Universidad Zaragoza
1542

# Trabajo Fin de Grado

# ESTUDIO DE LA INTEGRACION DE UN SISTEMA DE SEGUIMIENTO DE TROPAS EN EL FFT (FRIENDLY FORCE TRACKING)


Autor

Alejandro Menéndez Rillo

Director/es

Director académico: Dr. D. Jorge Ortín Gracia

Director militar: Cap. D. José Miguel Domínguez Rodríguez

Centro Universitario de la Defensa-Academia General Militar


Año 2016

## Resumen

El mundo en el que vivimos cambia con rapidez. En el ámbito civil, las empresas vuelcan sus esfuerzos en estar a la vanguardia en tecnología, en actualizarse y reinventarse constantemente. Una forma de conseguirlo es apostando por el software/hardware libre.

En el Ejercito de Tierra existe software de pago que permite ver la posición de grandes unidades en pantalla. Sin embargo, no existe ningún sistema de información que permita obtener la posición y el estado de un combatiente a pie en tiempo real.

En este Trabajo Fin de Grado se pretende resolver este problema mediante una solución basada en la plataforma electrónica Arduino. Este sistema es una plataforma de código abierto (open-source), basada en hardware y software flexibles y programables a la que se le puede añadir además diferentes sensores.

En concreto, en este Trabajo Fin de Grado se va a diseñar y elaborar un sistema que permita enviar a un nodo central en tiempo real la posición GPS de un dispositivo, así como otra información de interés (temperatura ambiental, pulso cardiaco) captada con sensores. Esta información se integra además en el sistema que usa actualmente el Ejercito de Tierra para realizar el posicionamiento de las unidades (sistema FFT).

## Abstract

The world is changing, and changing fast. The companies are trying to be at the forefront in technology in the civil field. Also, they are developing new strategies to reinvent themselves in the market. One way to get goals is using open source software and hardware.

In Spanish Army there is an expensive software to monitor and track the forces. This software provides information about the position of big units. It provides to the Army General information that help him in taking decisions. But this system is expensive, slow and very difficult to admin. In addition, there is no way to get information about one person.

This Final Degree Project develops an information system to provide information to Army General. This project uses the idea of future soldier, and it is developed using an open source electronics platform called Arduino. This board can be programed, also, we can add different type of sensors

In fact, Final Degree Project consists in design and making two prototypes. One of them will get information in real time about position (GPS), temperature, heart beat or other values. Then this information will be transferred to the other device. And this information will be integrated in FFT that is the system used by Spanish Army to track units.

In conclusion, this project solves the lack of information in the army using the technology.

# Agradecimientos

En este apartado, quiero dar las gracias a todos que en alguna manera han contribuido en mi formación tanto como ingeniero, como militar y sobre todo como persona.

Quiero agradecer la labor de los profesores, ya que sin vosotros esto no hubiera sido posible, gracias por haber estado allí en los momentos de mayor frustración, cuando no salían las cosas y de repente en la pizarra se solucionaban todos los problemas. Gracias por haber estado allí, día tras día, inculcándome unos estudios que al final se han convertido en una pasión. Además, extralimitándose en sus funciones muchas veces y ayudarme a madurar como persona.

Quiero agradecer a todos los profesores militares que he tenido por su dedicación y paciencia, los cuales aparte de enseñarme, me han inculcado unos valores muy importantes y que me servirán de estímulo en mi vida. Valores como abnegación, sacrificio y respeto, han hecho de mi ser mejor persona.

Quiero agradecer por su puesto, la ayuda prestada por el Regimiento de Transmisiones 21, sobre todo la del Capitán José Miguel Domínguez Rodríguez y la de los Tenientes Ramírez, Bayarri y Pariente que han sido todo un ejemplo para mí. También al Sargento Primero Tarín y a la Sargento Bordenca del aula CIS por sus conocimientos. Y a la Sección de radio de la 33 Compañía por su dedicación y paciencia.

También, quiero agradecer la estupenda labor realizada por el Doctor Jorge Ortín, sin el cual este trabajo no hubiera sido posible, muchas gracias por confiar en mí, haberme aconsejado y ayudado desde el principio.

Por último, quiero agradecer a mi familia y a Teresa la ilusión y el ánimo que me han transmitido, y sobre todo la paciencia que han tenido escuchando mis ideas, una y otra vez. Sin su apoyo, este proyecto no habría sido así.

# Contenido

# LISTA DE ILUSTRACIONES

## LISTA DE ACRONIMOS

| | CASTELLANO | INGLES |
|---|---|---|
| **AGM** | Academia General Militar | - |
| **BMS** | Sistema de Gestión del Campo de Batalla. | Battlespace Management Systems |
| **CIS** | Sistemas de Telecomunicaciones e Información | Communications and Information Systems |
| **COM** | Puerto de Comunicaciones en Serie | Communication Port |
| **ET** | Ejército de Tierra | - |
| **FFT** | Seguimiento de Fuerzas Propias | Friendly Force Tracking |
| **GPRS** | Servicio Radio de Conmutación de Paquetes | General Packet Radio Service |
| **GPS** | Sistema de Posicionamiento Global | Global Positioning System |
| **HF** | Alta Frecuencia | High Frequency |
| **IP** | Protocolo de Internet | Internet Protocol |
| **ISM** | Banda para uso industrial, Científico o Medico | Industrial, Scientific and Medical |
| **LED** | Diodo Emisor de Luz | Light-Emitting Diode |
| **LNA** | Amplificador de Bajo Ruido | Low Noise Amplifier |
| **NMEA** | Asociación de la Marina Estadounidense de Electrónica | National Marine Electronic Asociation |
| **OTAN** | Organización del Tratado Del Atlántico Norte | North Atlantic Treaty Organization |
| **PMET** | Publicación Militar del Ejercito de Tierra | - |
| **PR4G** | Programa Radio de Cuarta Generación | - |
| **QFD** | Despliegue de la Función de Calidad | Quality Function Deployment |
| **RBA** | Red Básica de Área | - |
| **RRC** | Red Radio de Combate | - |
| **RTP** | Red Táctica Principal | - |
| **SIM** | Módulo De Identificación de Abonado | Subscriber Identity Module |
| **SIMACET** | Sistema de Mando y Control del Ejército de Tierra | - |
| **SOTM** | Comunicación por Satélite en Movimiento | Satellite Communications On The Move |
| **SPI** | Interfaz de Control Serie de Periféricos | Serial Peripheral Interface |
| **UHF** | Frecuencia Ultra Alta | Ultra High Frequency |
| **USB** | Bus de Serie Universal | Universal Serial Bus |
| **UTP** | Par Trenzado sin Blindaje | Unshielsed Twisted Par |

# Capítulo 1. Introducción

Este Trabajo Fin de Grado (TFG) presenta el diseño y puesta en funcionamiento de un sistema que permite obtener en tiempo real la posición y el estado de un combatiente. Para desarrollar este sistema, se ha utilizado Arduino [1], que es una plataforma de hardware y software de uso libre. El propósito de este sistema es integrar la información recogida por varios sensores (posición GPS, pulso cardiaco, temperatura ambiental) en el programa de posicionamiento que usa el Ejército de Tierra, el FFT (*Friendly Force Tracking*). Con esta información se proveerá al Mando de herramientas de ayuda en la tarea de la toma de decisiones.

La elección de Arduino como base del sistema se fundamenta en la versatilidad que tiene esta plataforma, su bajo coste y su modularidad. La programación de un Arduino es sencilla, lo que permite cambiar o mejorar sus funciones con relativa facilidad. Además, permite añadir nuevos módulos o sensores en función de las capacidades que necesitemos.

## 1.1 Motivación del Trabajo

La principal motivación del trabajo es mejorar la información que dispone el Mando militar a la hora de tomar determinadas decisiones. Actualmente, en el Ejército de Tierra no existe ningún sistema que proporcione la información en tiempo real del estado de un combatiente (por ejemplo, su posición, su temperatura o su pulso). En este trabajo se propone un sistema viable para proporcionar al Mando información sobre la posición y el estado de su personal en tiempo real.

Además, el sistema propuesto hace uso de la propia infraestructura de telecomunicaciones que existe actualmente en el ejército y se integra en el software de posicionamiento que usa el Ejército de Tierra (FFT).

En el ejército, muchos teóricos hablan de los requisitos que debería tener el Combatiente del Futuro, algunos de estos requerimientos no se pueden realizar en este momento, debido a limitaciones tecnológicas, pero otros sí. La solución aquí planteada resuelve alguno de los problemas usando la tecnología actual, y demuestra que la idea del Combatiente del Futuro puede ser factible.

## 1.2 Objetivo y alcance del proyecto

El objetivo principal de este TFG es diseñar un dispositivo que pueda ser portado por una persona y que monitorice tanto su posición, a través de un módulo receptor GPS, como una serie de parámetros que pueden ser medidos a través de sensores. Posteriormente estos valores se transmiten a un nodo central y finalmente se integran en el sistema FFT. Este sistema se construirá en base a la plataforma de código abierto Arduino, explicada en el punto 2.3.

Para alcanzar este objetivo, se han definido unos requisitos que el dispositivo debe poseer:

- Debe ser de bajo coste y portátil.
- Debe tener suficiente autonomía.

- Debe tener suficiente cobertura para cubrir un despliegue de una sección de infantería ligera (1 kilómetro).
- No debe depender de la cobertura móvil de la red telefónica.
- Debe obtener la posición, temperatura, luminosidad y ritmo cardiaco.
- Debe integrarse en el sistema FFT.

En el Ejército de Tierra, actualmente existe un sistema de posicionamiento basado en los receptores GPS que llevan incorporadas las radios PR4G o HARRIS. No obstante, es un sistema relativamente lento, costoso y complicado de administrar. Además, la información que suministra se refiere únicamente a grandes entidades, a partir de sección (unas treinta personas). Esto hace que la información sea parcial, puesto que se sabe la posición de la entidad al completo, pero no la de cada combatiente en concreto. Por el contrario, el sistema que se propone en este TFG trabaja a nivel de cada persona, por lo que se puede llegar a conocer dónde y cómo se encuentra cada soldado desplegado en el terreno de batalla.

El desarrollo del sistema se habrá completado cuando se haya obtenido un prototipo que cumpla los requisitos y que además integre los valores obtenidos de los sensores en el programa FFT.

## 1.3 Ámbito de aplicación

Una vez desarrollado este sistema, las posibilidades de uso son múltiples gracias a su modularidad y versatilidad. En el ejército, una de las aplicaciones que podría tener es la de conocer la localización de todos y cada uno de los miembros de una patrulla de reconocimiento de Operaciones Especiales, o de los paracaidistas que son lanzados en una zona. Además de saber su estado gracias a los sensores incorporados.

De hecho, las aplicaciones de este sistema no solo se centran en el Ejercito de Tierra, si no que se podría instalar en efectivos de la Unidad Militar de Emergencias, policías, bomberos o Grupos de Rescate e Intervención en Montaña de la Guardia Civil.

Fuera del entorno puramente militar, el sistema permitiría a un médico controlar el estado de una persona sin que tuviera que desplazarse al centro sanitario, puesto que se obtienen los valores de temperatura y pulso. También sería de aplicación para el cuidado de ancianos o niños, ya que permitiría saber dónde y cómo se encuentran estas personas. Al ser modular, se le pueden instalar diferentes sensores para cumplir con diferentes necesidades.

## 1.4 Metodología

El diseño y montaje del prototipo se ha realizado en las siguientes etapas:

En un primer momento, se analizaron las capacidades que debía ofrecer el sistema y se realizó una búsqueda y adquisición de los componentes electrónicos necesarios para poder ofrecer dichas capacidades. En el Anexo A se muestra el presupuesto de los componentes adquiridos.

Seguidamente se realizó la conexión física de los Arduinos con los sensores, la pantalla y los módulos de comunicación NFR24L01, comprobando que funcionaban por separado. Para ello, fue necesario escribir en el lenguaje de programación C++ el código necesario para controlar los componentes (sensores, pantalla y módulo de comunicación) a través de Arduino [2]. Testando los componentes por separado se asegura que funcionen correctamente y se aíslan los posibles errores que pudieran surgir por incompatibilidad entre ellos.

La siguiente fase fue montar todos los componentes simultáneamente para comprobar su funcionamiento, modificando el código de programación para que los módulos y sensores trabajen a la vez. Esto implica la recogida de la información suministrada por los sensores y su posterior transmisión a través de los módulos de comunicación.

Finalmente se desarrollaron una serie de pruebas para verificar la viabilidad de integrar los datos recogidos por el sistema en el software FFT. Además, con estas pruebas se demostraron las capacidades que posee el sistema.

## 1.5 Estructura de la memoria

La memoria está organizada de la siguiente forma:

- En el Capítulo 2 se expone un análisis del estado del arte, además de una introducción a la plataforma Arduino y a los diferentes sistemas que se usan en el Ejército de Tierra para el Mando y Control.
- El Capítulo 3 se centra en el desarrollo del sistema, la programación y la integración del sistema propuesto en el FFT y SIMACET[1].
- El Capítulo 4 expondrá las diferentes pruebas realizadas para demostrar las capacidades del sistema, alcance, autonomía y compatibilidad.
- En el Capítulo 5 se muestran las conclusiones finales y las líneas futuras.

Al final del documento se encuentran los siguientes anexos:

- En el Anexo A se muestra el coste real de los componentes utilizados para montar los prototipos.
- En el Anexo B se realiza una presentación del prototipo construido, además de unas fotografías mostrando los componentes electrónicos.
- En el Anexo C se realiza un estudio de mercado usando la herramienta de la casa de la calidad del QFD[2] (*Quality Function Deployment).*
- En el Anexo D se explica qué es la codificación NMEA[3] y la aplicación en el prototipo.

[1] Sistema de Mando y Control del Ejército de Tierra
[2] Herramienta usada en diseño de productos que establece relaciones entre los deseos de los clientes y las capacidades a desarrollar en los productos.
[3] NMEA (National Marine Electronic Asociation) es un protocolo usado para la navegación tanto marítima como terrestre.

- En el Anexo E se muestran las especificaciones del módulo de comunicaciones NRF24L01.
- En el Anexo F se muestran unas fotografías del sistema funcionando en los vehículos del Regimiento de Transmisiones 21 de Marines (Valencia).
- Finalmente, en el Anexo G se muestra el código completo de programación utilizado los prototipos.

# Capítulo 2. Estado del arte

## 2.1 Introducción.

El Mando y Control se define como el ejercicio de la autoridad, la conducción y seguimiento de las fuerzas asignadas para el cumplimiento de la misión por el Mando operativo expresamente designado. El Mando desarrolla las funciones de Mando y Control a través de los Sistemas de Mando y Control[4]. Así el Mando y Control no solo se refiere a los procesos, si no que necesita de sistemas de información que permitan al Mando tomar decisiones en base a estos.

La información para ser válida debe ser contrastada, actualizada y en tiempo real. Tener una información de calidad permitirá mantener una posición ventajosa en el terreno de combate frente al enemigo. Es muy importante no olvidar el componente de seguridad y los problemas que pueden surgir, si parte de esta información es filtrada al enemigo. Aunque existen múltiples soluciones para evitar las fugas de información, nunca hay que descuidar las transmisiones.

Dentro del ejército español, las Unidades de Transmisiones del Ejército de Tierra están organizadas para establecer, operar y mantener los CIS (*Communications and Information Systems*, sistemas de telecomunicaciones e información), y así articular el Sistema de Mando y Control del Ejército de Tierra (SIMACET). Para ello, cuentan con la red de comunicaciones permanentes y una táctica (véase Ilustración 1).

En el nivel de comunicaciones permanente, se encuentra la Red Táctica Principal (RTP), con una red y nodos desplegados en territorio nacional. El siguiente nivel es el nivel de comunicaciones táctico y se divide en dos subniveles, la Red Básica de Área (RBA) y la Red Radio Combate (RRC). La RBA genera una red mallada no permanente, en base a estaciones de comunicación móviles. En cambio, en la RRC los sistemas de comunicación son de menor capacidad y portátiles. Todas estas redes se pueden interconectar mediante elementos cableados o por comunicación satélite.

[4]Extraido del Field Manual 6-0 Mission Command: Command and Control of Army Forces, Headquartes United States Army, Department of the Army, Washignon DC 2003

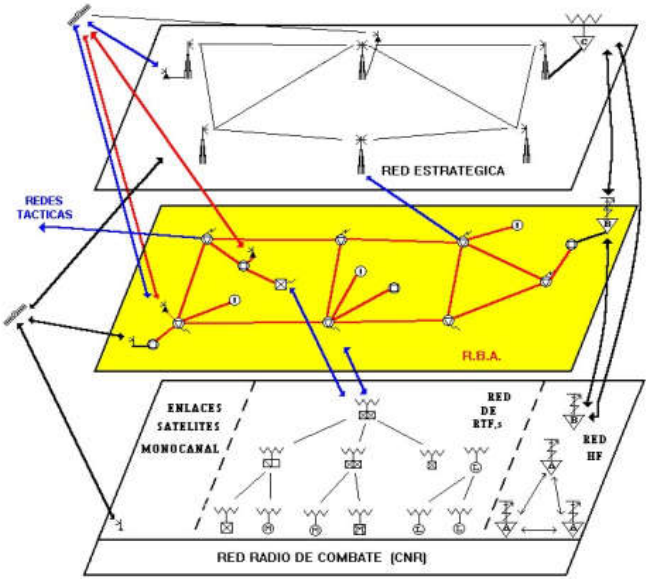



**Ilustración 1 - Sistema de comunicaciones ET[5]**

El sistema propuesto se encuadra fundamentalmente en la RRC, ya que es el principal medio de comunicaciones usado en los escalones de brigada e inferiores, tipo Compañía, Sección, Escuadra y Pelotón. La RRC trabaja normalmente en HF, VHF, y UHF, aunque últimamente gracias a los nuevos sistemas SOTM (*Satellite Communications On The Move)*, está disponible la comunicación satélite a nivel Compañía mientras el vehículo se está moviendo.

## 2.2 Sistemas de Mando y Control para pequeñas unidades

Los sistemas de Mando y Control se pueden dividir según la envergadura de la unidad para la que son diseñados, así tenemos sistemas para Gran Unidad o Pequeña Unidad. Esta división es equivalente a la que se podría hacer entre sistemas de Mando y Control de nivel estratégico y de nivel táctico. A nivel Gran Unidad se encuentra el sistema SIMACET.

La aplicación del sistema propuesto se enmarca en los sistemas de Mando y Control a nivel táctico o de Pequeña Unidad, donde nos podemos encontrar diversos sistemas tales como el BMS (*Battlespace Management System*) o el FFT citado en el capítulo anterior. Puesto que de estos dos sistemas el más usado en el Ejército es el FFT, el dispositivo desarrollado en este TFG va a integrarse en él.

[5] Fuente: PMET OR3-501

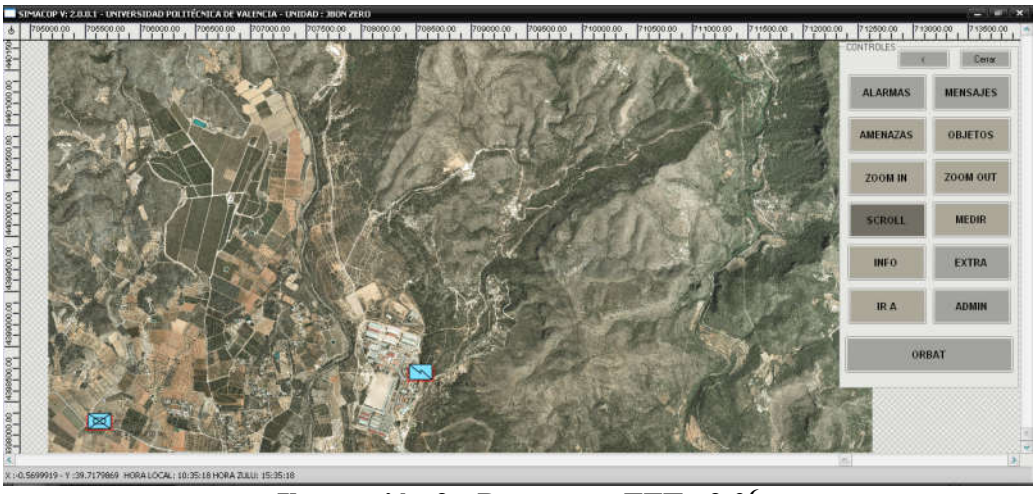



**Ilustración 2 - Programa FFT v2.0[6]**

El sistema FFT comprende desde vehículo de pelotón hasta Batallón (véase Ilustración 2). Este sistema intercambia información a través de los medios radio llevados en el vehículo, ya sea a través de radios PR4G o HARRIS que forman parte de la RRC. Además, proporciona diversas herramientas al operador, como envío de mensajes, envío de alarmas, establecer objetos sospechosos en el mapa, información extra sobre las unidades y medición de distancias.

No obstante, el sistema FFT es un sistema pensado para posicionar vehículos y no para el posicionamiento del combatiente individual. Adicionalmente, no permite obtener datos relativos al estado físico de los combatientes.

Otro problema que presenta es que es relativamente lento en el envío de información y no se puede considerar un sistema basado en tiempo real ya que pueden pasar hasta 5 minutos desde que se genera la información hasta que está disponible para todos los usuarios del sistema.

Por el contrario, la solución propuesta permite obtener la posición, la temperatura, el pulso y la luminosidad de un combatiente a pie en tiempo real. Además, se va a integrar esta información en el sistema FFT ya que, pese a sus limitaciones, es el sistema empleado en la actualidad.

## 2.3 Plataforma Arduino

A continuación, se presenta una breve presentación de la plataforma Arduino, la cual se ha usado como base para desarrollar el dispositivo desarrollado en este TFG. Arduino es una plataforma de código abierto para realizar prototipos de electrónica. Está basada en un microcontrolador[7] ATMEL®, el cual se puede programar para interactuar con otros elementos electrónicos. Posee diferentes entradas y salidas tanto analógicas como digitales, necesarias para comunicarse con los sensores, actuadores o incluso un modelo Arduino con otro. Existen diferentes modelos en el mercado dependiendo del uso que le vayamos a dar (véase Ilustración 3) [1].

[6] Fuente: Elaboración propia.
[7] Un microcontrolador o microchip es un circuito integrado en el cual se pueden grabar instrucciones.

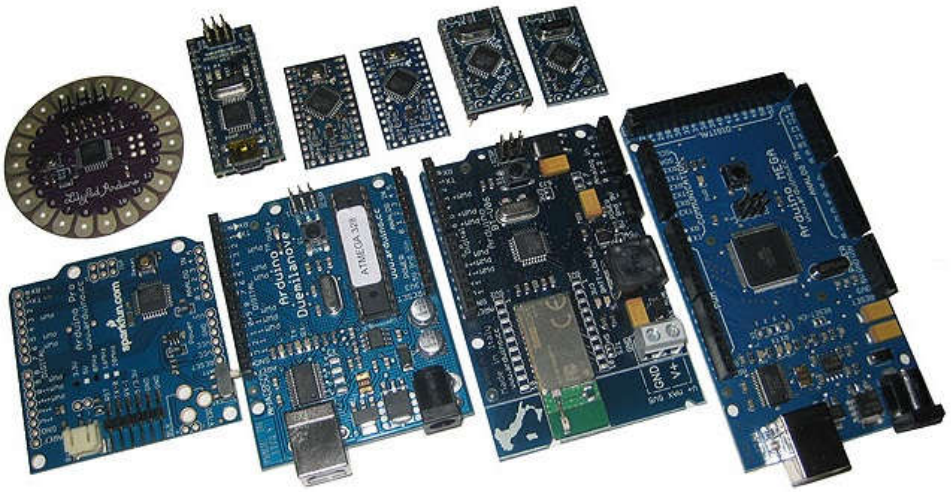


**Ilustración 3 - Tipos de microcontroladores[8]**

Todas estas placas electrónicas poseen un microcontrolador ATMEL®, el cual es el "cerebro" del Arduino. Este chip es programado y es el encargado de realizar las operaciones lógicas y matemáticas. Además, la placa integra un cristal oscilador, su trabajo es darle al microchip un pulso constante, en definitiva, la frecuencia de reloj. En concreto, los modelos elegidos que se van a emplear en este proyecto son el *Arduino Uno - R3* y el *Arduino Mega 2560*. La razón de haber elegido estos dos modelos es que se ajustan perfectamente a lo requerido para el sistema. El Arduino *Uno - R3* se ha usado como dispositivo local, porque es pequeño, económico y versátil. En el dispositivo móvil se ha optado por el *Arduino Mega 2560,* debido a que, a pesar de ser más grande, posee mayor cantidad de entradas y salidas, (tanto de carácter analógico como digital), por ende, se pueden conectar mayor número de sensores.

Lo realmente interesante de estas placas es que permiten interactuar con el exterior a través de sus sensores analógicos o digitales. Existen muchos tipos de sensores para diversas funciones, (véase Ilustración 4): detectores de humo, de luminosidad, de infrarrojo, pulsadores, cámaras, brújulas, inclinómetros, acelerómetros, de nivel de líquido, de humedad, de temperatura, de presencia, módulos de comunicación WiFi, Bluetooth …

[8] Fuente: [1]


**Ilustración 4 - Sensores de Arduino[9]**

La placa Arduino se puede programar conectándola a un ordenador a través del puerto USB. El lenguaje de programación utilizado es un derivado del C, aunque admite instrucciones de C++ [2]. A modo de resumen, el cuerpo del programa está formado por estas dos funciones principales:

```
void setup () { //Se ejecuta una sola vez cuando el controlador se enciende. Se usa para programar las variables iniciales.

}

void loop () {  //Es un bucle infinito. Leerá los sensores y tomará decisiones.

}
```

Es posible encontrar infinidad de aplicaciones realizadas sobre Arduino. Por ejemplo, en investigaciones científicas se pueden emplear para recoger, procesar y enviar valores analógicos o digitales (temperatura, humedad, presión…). En el campo de la domótica, permite controlar elementos de la casa como son las persianas o detectar una presencia intrusa. En robótica usando motores paso a paso controlados por el Arduino. Otros ejemplos son: *Ardupilot*[10] (software y hardware de aeronaves no tripuladas), máquinas de control numérico por computadora (CNC), impresoras en 3D… En resumen, Arduino es un sistema muy versátil que se puede emplear en cualquier situación en la que se necesite controlar de modo autónomo algún sistema mecánico o realizar medidas con sensores.

El sistema que se propone, está basado la plataforma Arduino, de software y hardware libre. Es cierto que, dentro del Ministerio de Defensa de España existe

[9] Fuente: AMAZON
[10] Ardupilot: http://www.ardupilot.com/

cierta reticencia al uso de aplicaciones de código abierto[11] o de plataformas libres, debido a que se consideran inseguras, al no estar certificadas por el Centro Criptográfico Nacional. Sin embargo, poco a poco en los ordenadores del ejército se pueden encontrar programas como el *Open Office*[12] o el programa *Asterix*[13] instalado en un sistema operativo basado en *Linux*[14], los cuales son de libre distribución. Por tanto, el uso de este sistema no sería tan excepcional.

[11] Es software cuyo código fuente u otros derechos son publicados bajo una licencia de libre uso o forman parte del dominio público.
[12] Conjunto de aplicaciones ofimáticas de código abierto, como procesador de textos, hoja de cálculo o de presentaciones.
[13] Centralita telefónica de Voz sobre IP.
[14] Existen sistemas operativos libres que están basados en este código. Ubuntu, Gentoo, Debian…

# Capítulo 3. Diseño del sistema

## 3.1 Descripción detallada del núcleo del proyecto

En este capítulo se va a explicar detalladamente el funcionamiento del sistema propuesto. En primer lugar, el dispositivo remoto (Arduino portado por un soldado) obtiene la información de los sensores de temperatura, luminosidad, pulso y posición GPS. Esta información se transmite a un dispositivo local (otro Arduino) a través de una conexión inalámbrica suministrada por los módulos de comunicación NRF24L01[15]. Una vez que se recibe la información, se reenvía mediante un cable USB a un ordenador. Este ordenador tiene instalado el sistema FFT e integra la posición del dispositivo remoto en el mismo (la posición del dispositivo remoto aparece en el mapa que presenta FFT por pantalla).

Toda la información se puede mandar también a través de la radio PR4G/HARRIS a un servidor SIMACET para que esté disponible al Mando (véase Ilustración 5).

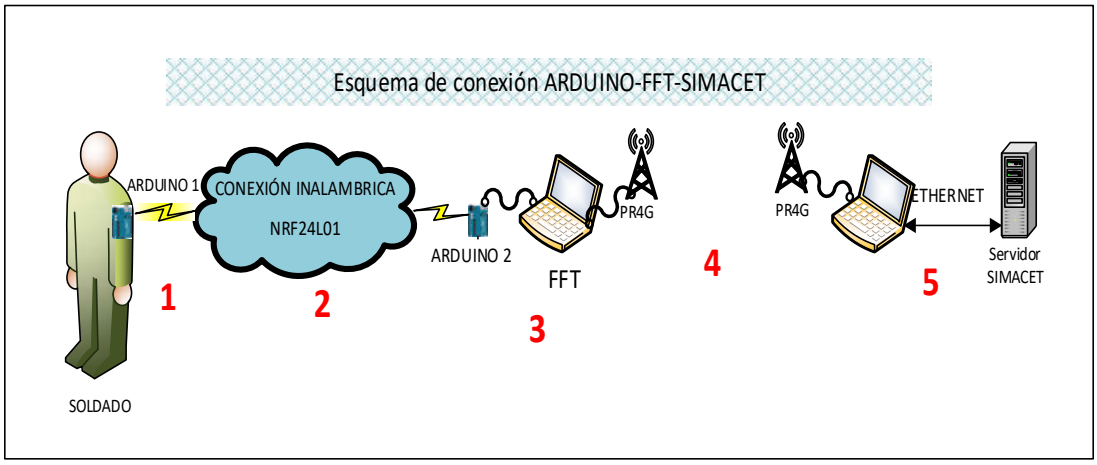



**Ilustración 5 - Esquema general.[16]**

Para el uso táctico del sistema (véase Ilustración 6), cada soldado portaría un Arduino que correspondería al dispositivo remoto citado en el párrafo anterior. Este Arduino envía información a otro Arduino (dispositivo local) y este a su vez a un ordenador que tuviera el FFT. Tanto este Arduino como el ordenador estarían ubicados en un vehículo.

El sistema admite hasta 6 dispositivos conectados en estrella a un nodo central. Los diferentes vehículos establecerían el enlace a través de radios de mayor alcance

[15] Véase Anexo E para consultar especificaciones del módulo de comunicaciones nRF24L01.
[16] Fuente: Elaboración propia.

hacia el escalón superior. Por ultimo existiría un servidor con SIMACET el cual almacenaría las posiciones de todo el personal desplegado en el terreno de combate.

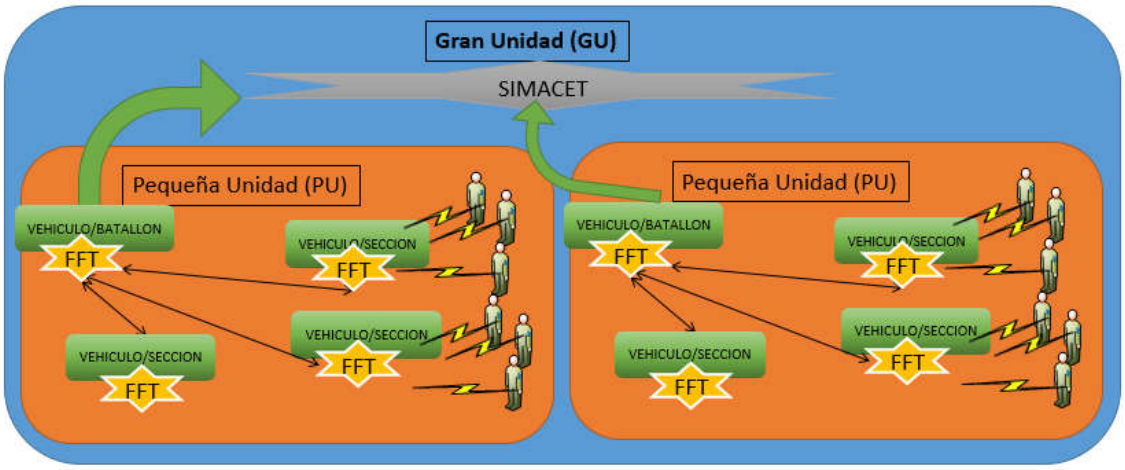



**Ilustración 6 - Uso del Arduino en entorno táctico.**[17]

## 3.2 Programación Arduino local (en el vehículo)

En este apartado se mostrará a rasgos generales cómo se ha realizado la programación en el Arduino que está ubicado en el vehículo. A nivel hardware está compuesto por la placa Arduino UNO, un LCD, dos pulsadores y el módulo radio. En el Anexo B se encuentra el esquema de conexiones. Como se ha indicado anteriormente, los equipos se han programado usando el lenguaje de programación C++ [2]. El código programado se consta de tres bloques: un bloque inicial de inclusión de librerías y de declaración de variables globales, un bloque correspondiente a la función setup(), la cual se ejecuta una sola vez al inicializar el dispositivo, y un último bloque correspondiente a la función loop(), que se ejecuta continuamente en un bucle infinito (véase Ilustración 7).

En el caso concreto del Arduino local se ha realizado la siguiente programación: en la parte inicial del código de programación se han incluido las librerías necesarias para que los módulos funcionen, además se han declarado las variables globales que estarán disponibles para las demás funciones. En la función setup() se configura el LCD y el modulo radio (velocidad, tamaño paquete, direcciones, pines y canal). Además, se establece la comunicación con el ordenador a través del puerto serie. En la función loop() se leen los valores recibidos del Arduino remoto, se muestra en el LCD la información de los sensores formateada correctamente, se muestran esos valores también en la pantalla del ordenador y por último se envía, si se desea, un mensaje al Arduino remoto. En el Anexo G se puede encontrar el código completo de programación utilizado.

[17] Fuente: Elaboración propia.

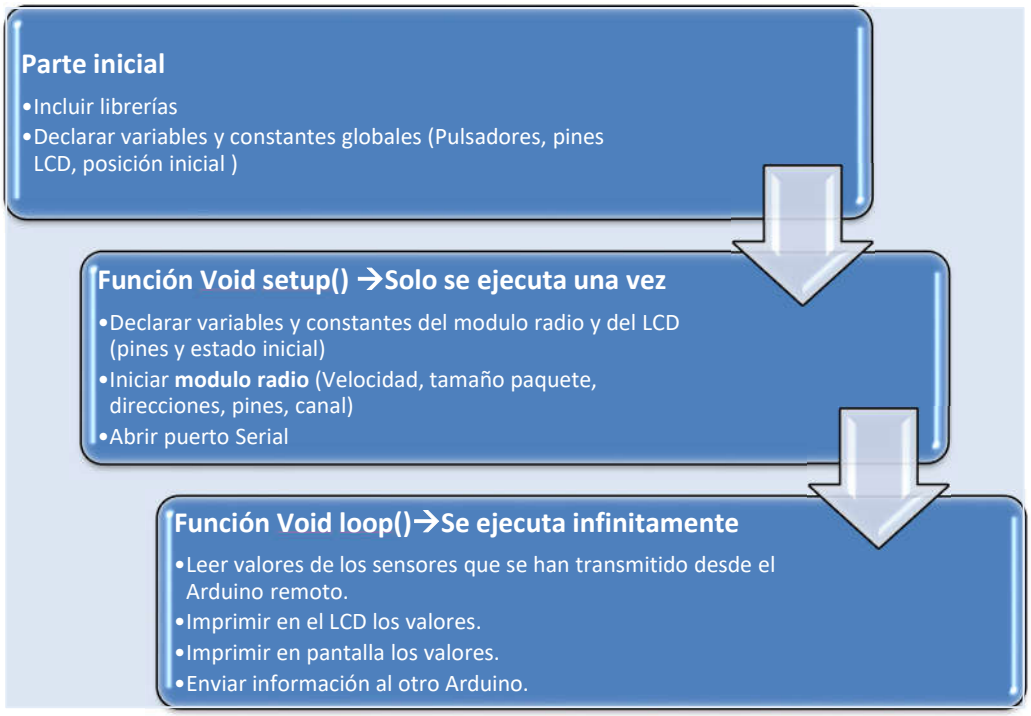



**Ilustración 7 - Programación en alto nivel del Arduino local.[18]**

### 3.2.1 Módulos de comunicación NRF24L01

Para el establecer la comunicación entre los dos dispositivos se han utilizado dos módulos de radio NRF24L01. A continuación, se detallan las características más importantes extraídas de la hoja de especificaciones del producto[19].

El NRF24L01 es un chip radio transceptor que funciona en la banda ISM de 2.4GHz. El modulo consiste en un sintetizador de frecuencias, un amplificador, un oscilador de cuarzo, un modulador, un demodulador, una antena y varios sistemas para asegurar la comunicación.

El modulo es programable y configurable a través de la interfaz SPI (Serial Peripheral Interface) del Arduino. Para el manejo de estos transceptores, se dispone de las librerías NRF24 y MIRF. Al evaluar ambas librerías durante la ejecución del TFG se observó que la librería NRF24 presentaba problemas de transmisión, por lo que finalmente se ha optado por usar la MIRF. Esta librería admite hasta 6 dispositivos conectados a un nodo central.

La transmisión con los módulos NRF24L01 emplea un protocolo propio, el Enhanced ShockBurst™, basado en comunicación por paquetes. La transmisión se realiza usando modulación GFSK (Gaussian Frecuency Shift Keying).

La velocidad máxima de transmisión es de 2Mbps y su bajo consumo (40mW cuando se transmite a 0dBm), lo hace perfecto para comunicaciones de corto alcance. La distancia de transmisión depende de las condiciones del terreno, pero la media es

[18] Fuente: Elaboración propia.
[19] Véase Anexo E para consultar especificaciones del módulo de comunicaciones nRF24L01.

de 100 metros. No obstante, en el proyecto se van a usar unas antenas de mayor ganancia para conseguir mayor alcance.

En el Anexo E se describen más detalladamente las características técnicas de estos módulos. Los módulos se muestran también en el Anexo F de fotografías.

Las placas Arduino no solo admiten este tipo de módulos de transmisión. De hecho, la comunicación se podría haber realizado a través de conexión WiFi o Bluetooth. Ambas tecnologías se han probado también, si bien se han descartado ya que el alcance era muy inferior (apenas 20 metros). Otra opción considerada fue la posibilidad de usar conexión GPRS, pero también se ha descartado debido a los siguientes motivos: (i) el coste de los módulos GPRS asciende a unos 60€ por unidad, (ii) requieren una tarjeta SIM y un contrato con una compañía telefónica, y (iii) el sistema dependería de la cobertura suministrada por la compañía telefónica, por lo que el dejaría de ser autónomo y no podría ser usado ser usado en emergencias o en situaciones tácticas en el extranjero.

## 3.3 Programación Arduino remoto

En este apartado se incluye la programación a alto nivel del Arduino remoto cuyo esquema de bloques aparece en la ilustración 8.

Este Arduino es diferente al local puesto que consta de los sensores de posición (GPS), luz, temperatura y pulso, además del módulo radio y de la pantalla LCD con los pulsadores. En el Anexo B se encuentra el esquema de conexiones.

En el código de programación, en la parte inicial se ha tenido que tener en cuenta la declaración de variables de estado relativas a los sensores. Posteriormente, en la función setup() se han inicializado todos los sensores y se han configurado. Para la función en bucle loop(), se ha tenido que programar el Arduino para que lea los valores obtenidos a través de los diferentes sensores y los imprima en pantalla y en el LCD en un formato correcto para su lectura. Finalmente, esos valores son transmitidos hacia el otro Arduino. Además, se muestra en el LCD el posible mensaje que haya llegado desde el Arduino local. En el Anexo G se puede encontrar el código completo de programación utilizado.

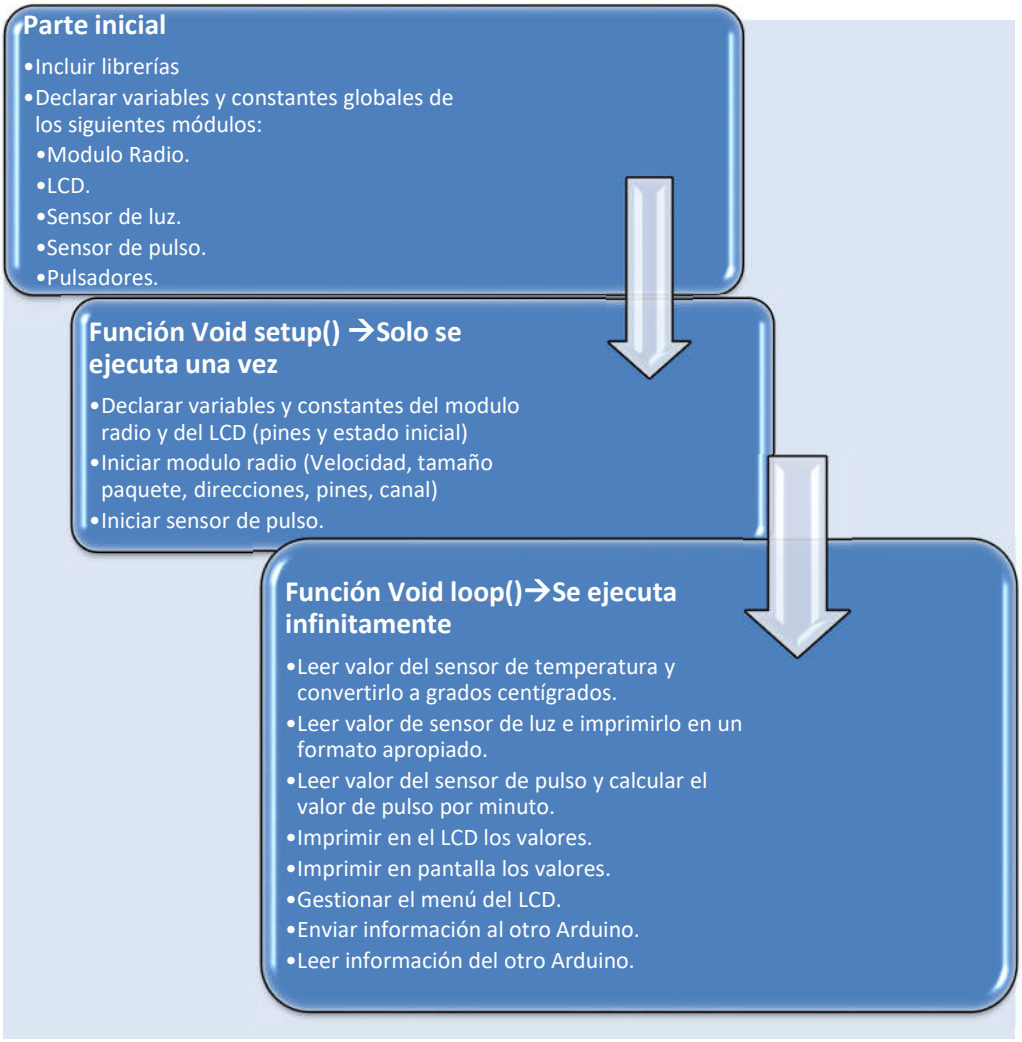



**Ilustración 8 - Programación en alto nivel del Arduino remoto.**[20]

A continuación, se explican las especificaciones de los distintos sensores que lleva el Arduino remoto.

### 3.3.1 Módulo GPS Ublox NEO - 6M

El prototipo obtiene la posición GPS gracias al Módulo GPS Ublox NEO - 6M, (Ilustración 9). Este dispositivo es compatible con el protocolo NMEA [3] (descrito en el Anexo D). Una vez conectado y configurado correctamente proporciona la posición GPS y otros datos como la velocidad, la altura o la aceleración[21].

[20] Fuente: Elaboración propia.
[21] La librería utilizada es la TinyGPS++, descargable desde http://arduiniana.org/libraries/tinygpsplus/ (Consultado el 27 de septiembre de 2016).

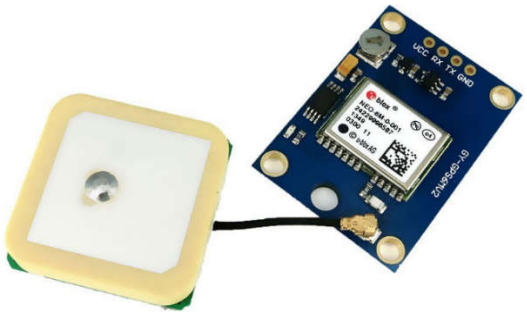


**Ilustración 9 - Módulo GPS Ublox NEO - 6M[22]**

El Arduino remoto está programado de tal modo que recoge la sentencia NMEA generada por el módulo (con información de la posición) y la transmite al Arduino local que se encuentra en el vehículo. Posteriormente se ha realizado un programa que permite que esta sentencia NMEA sea reconocida por el sistema FFT para poder mostrar la posición del soldado actualizada en el mapa.

### 3.3.2 Sensores

#### *3.3.2.1 Térmico*

Para obtener el valor de la temperatura se ha instalado un sensor de temperatura digital, en concreto el LM35. Este módulo proporciona una salida de voltaje proporcional a la temperatura. La salida es lineal y aumenta en 10mV con cada grado centígrado. El rango es de -55ºC (-550mV) a 150ºC (1500 mV) y su precisión es de 0,5ºC.

#### *3.3.2.2 Fotoresistor*

Para medir en nivel de luz se ha instalado una fotorresistencia GL55, que es una resistencia variable en función de la luz recibida. Leyendo el valor del voltaje a través de las entradas analógicas podemos estimar la cantidad de luz ambiental. Los valores de la resistencia son 1 Mohm en total oscuridad y unos 100 Ohm bajo luz brillante. Este dispositivo es bastante sensible a las variaciones de temperatura, pero para un prototipo es suficiente.

#### *3.3.2.3 Pulsómetro*

Para obtener el pulso se ha instalado un sensor de pulso fabricado por http://pulsesensor.com/ el cual se puede poner en el dedo o lóbulo de la oreja. Cuenta con una aplicación de código abierto que muestra en tiempo real la gráfica de la frecuencia cardíaca. La tensión de trabajo es de 3.3V a 5V.

#### *3.3.2.4 Batería de 2200mAh*

Para alimentar el dispositivo móvil se ha optado por usar una batería recargable de 2200mAh a 5V que proporciona una salida de 1A. Esta batería nos brinda una autonomía aproximada de 8 horas conectada al puerto USB.

[22] Fuente: AMAZON

## 3.4 Integración con el programa FFT.

Como se ha visto en los capítulos 1 y 2, el ejército español usa el sistema FFT para realizar la localización de las unidades. No obstante, este sistema no está preparado para hacer el seguimiento de personal a pie ya que precisa de unos sistemas de transmisión que son demasiado pesados para que un soldado los pueda portar con facilidad. Gracias al sistema propuesto en este TFG se puede realizar esta localización e incluir la información recogida por los sensores. El siguiente paso es integrar la información recogida por el dispositivo remoto en el sistema FFT.

El sistema FFT representa la posición de una unidad en el mapa mediante unas trazas que recibe a través de los medios radio de la Unidad (cada radio dispone de un receptor GPS). El sistema FFT es bastante rígido y únicamente acepta la información suministrada por las radios PR4G y HARRIS o por un GPS externo (véase Ilustración 10).

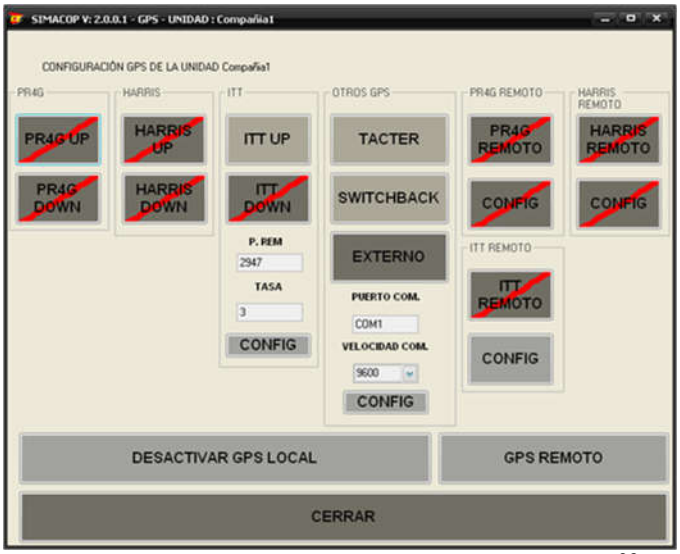



**Ilustración 10 -Configuración GPS en FFT[23].**

Debido a que no se dispone el código fuente del sistema FFT, no se puede modificar para que acepte la información suministrada por el Arduino directamente. A pesar de ello, se ha conseguido representar la posición del dispositivo remoto en el sistema FFT configurando el programa para que lea la información GPS a través del puerto COM.

Para ello, se conecta a través de un cable UTP un ordenador portátil (Portátil 2 en la Ilustración 11) a otro ordenador portátil (Portátil 1). Puesto que el sistema FFT acepta la conexión de un GPS externo a través del puerto COM, se configura el Portátil 2 para que acepte tramas GPS a través del puerto USB del Arduino, programándose el Arduino para que mande todo lo que le llegue a través del puerto COM.

[23] Fuente: Elaboración propia.

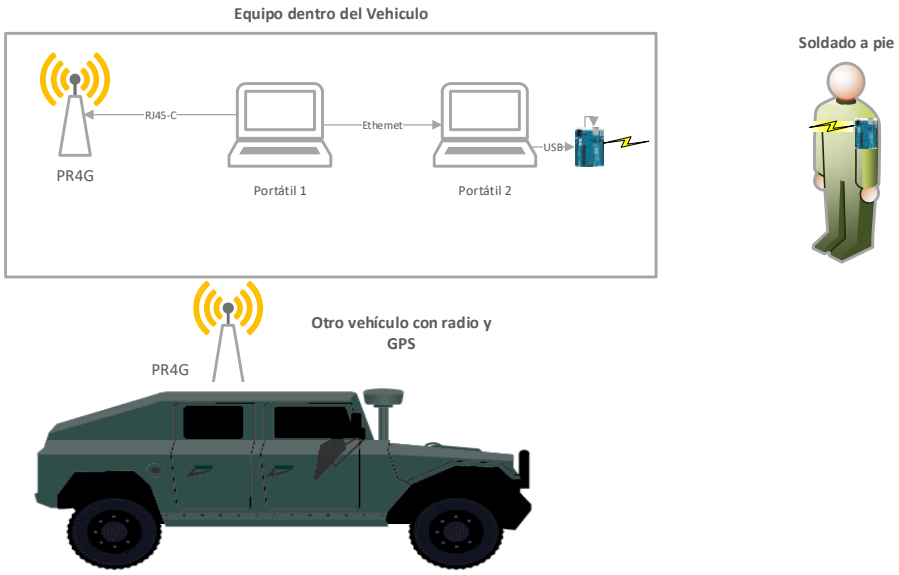



**Ilustración 11 - Esquema de pruebas con FFT.[24]**

## 3.5 Integración con el sistema SIMACET a través del diodo de datos.

Como se ha explicado anteriormente, el Sistema de Mando y Control del Ejército de Tierra (SIMACET) se usa para establecer la posición de las Grandes Unidades. Sin embargo, es posible que al Mando le interese saber la posición de un soldado determinado.

Las posiciones presentes en el sistema FFT se pueden integrar en SIMACET a través de un diodo de datos (Ilustración 12). Este diodo de datos sirve para poder integrar un sistema no seguro, como es FFT, con un sistema seguro como SIMACET. La integración se realiza mediante el programa Antares, el cual permite que las posiciones establecidas por el sistema FFT aparezcan también en SIMACET (Ilustración 13).

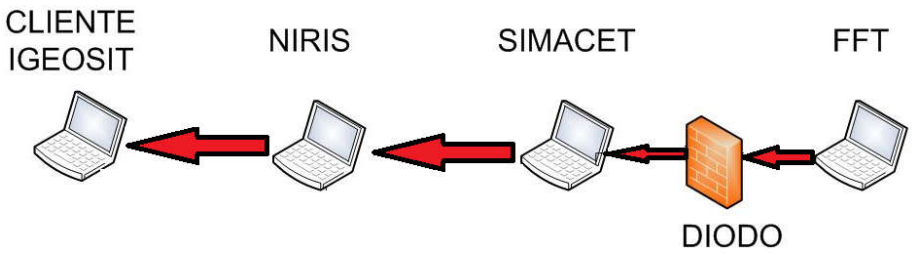



**Ilustración 12 - Esquema de conexión del FFT a una red OTAN[25][26].**

[24] Fuente: Elaboración propia.
[25] El software IGeoSIT, es una herramienta elaborada por la Agencia de Comunicación e Información de la OTAN, para mostrar situaciones de alarma en un mapa.
[26] Fuente: Elaboración propia.

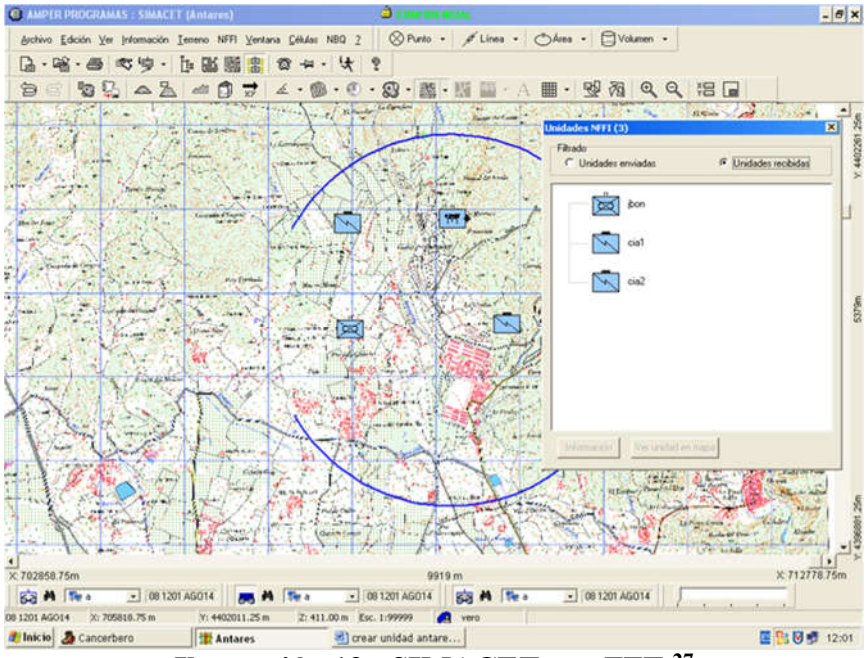



**Ilustración 13 - SIMACET con FFT.**[27]

[27] Fuente: Elaboración propia.

# Capítulo 4. Pruebas

Para demostrar las capacidades del sistema se han realizado una serie de pruebas que se describen en este capítulo.

## 4.1 Materiales y herramientas

Para la realización del prototipo y las pruebas se han utilizado los siguientes elementos:

Hardware:

- Arduino UNO.
- Arduino MEGA con sensores de ritmo cardiaco, de temperatura LM35, de luz, GPS NEO 6M
- 2 Módulos de comunicación NRF24L01.
- Antena de 3dBi y antena de 6dBi.
- 2 Radios PR4G (VHF)
- Ordenador portátil.

Software:

- Kit de desarrollo Arduino.
- Programa FFT desarrollado por la Universidad Politécnica de Valencia

## 4.2 Pruebas de alcance en exterior

El propósito de estas pruebas es valorar el alcance del dispositivo en condiciones controladas variando la configuración interna del mismo.

El dispositivo puede funcionar a máxima potencia de transmisión (0dBm) en dos velocidades (1 Mbps, 2 Mbps), adicionalmente se puede añadir un amplificador de bajo ruido (LNA[28]). Para configurar estos parámetros, se debe modificar el registro RF_SETUP de los módulos de comunicaciones. Es importante poner los mismos valores en los dos dispositivos (transmisor y receptor) ya que en caso contrario el sistema no funcionará.

Las pruebas se realizaron en un descampado de la Base General Almirante, situada en Marines (Valencia), con coordenadas GPS (39.712217,-0.5703535), durante el periodo de tiempo comprendido del 10 de octubre al 14 de octubre de 2016 (véase Ilustración 14).

[28] LNA: Low Noise Amplificator, es amplificador electrónico usado en receptores donde existe una gran presencia de ruido, la función es de incrementar la señal y eliminar el ruido electromagnético que degrada la señal.

**Ilustración 14 - Zona de pruebas dentro del cuartel de Marines.** [29]

El Arduino transmisor emplea una antena de ganancia 3dBi. Al receptor se le ha cambiado la antena de origen por una de mayor ganancia (6dBi). Se considera que el alcance ha llegado a su distancia máxima cuando la trama GPS llega incorrectamente a la estación base.

Los resultados obtenidos han sido los siguientes:

| Velocidad | LNA ACTIVO | Distancia (metros) |
|---|---|---|
| **2Mbps** | SI | **135** |
| **2Mbps** | NO | **62** |
| **1Mbps** | SI | **157** |
| **1Mbps** | NO | **75** |

**Tabla 1 - Resultados de alcance.**

Se puede comprobar que al desactivar el LNA perdemos alcance. Según la hoja de características, al desactivarlo perdemos 1,5 dB en recepción, pero ahorramos 3mW en gasto energético.

En este caso, el ahorro energético no compensa la pérdida de ganancia en recepción. Por ello, para el resto de pruebas se mantendrá el LNA activado. Dado el volumen de datos que es necesario transmitir, la velocidad seleccionada debería ser de 1 Mbps y en caso de querer transmitir video, se debería aumentar la velocidad a

[29] Fuente: GOOGLE MAPS

2 Mbps a pesar de la disminución en alcance. El Anexo F contiene más documentación gráfica de cómo fue realizado la prueba.

## 4.3 Pruebas de alcance en interior

Una vez fijados los parámetros de transmisión óptimos en exteriores, se realizan pruebas en el interior de un edificio con la máxima potencia de transmisión (0dBm), dos velocidades (1Mbps, 2Mbps) y el LNA activado.

Tras realizar las pruebas, se ha conseguido un alcance de unos 40 metros para ambas velocidades. De todos modos, este resultado no es concluyente debido a la influencia que tiene el tipo de construcción donde se realicen las pruebas sobre la transmisión inalámbrica. Por ejemplo, valores como el grosor de las paredes o el tipo de materiales influyen enormemente en la atenuación de la señal.

## 4.4 Pruebas de conectividad en el FFT

Para comprobar la fiabilidad del sistema FFT y su tiempo de respuesta, se efectuó una prueba del mismo en condiciones reales y sin integrar el sistema diseñado con los Arudinos. Esta prueba fue realizada en las maniobras TIWAR 2016 ejecutadas en el Regimiento de Transmisiones 21 de Marines (Valencia) durante la semana del 26 a 30 de octubre de 2016.

El programa que se utilizó es el FFT 2.0, el cual es solo compatible con Windows XP. Para el test de conectividad se procedió a instalar el programa en dos máquinas virtuales con Windows XP SP4 sobre un mismo PC. El programa de virtualización empleado fue el VirtualBox 5.1.6 de Oracle. Ambas máquinas estaban conectadas mediante un switch virtual interno, por lo que en esta casa la red se puede considerar ideal (sin retardo y con un ancho de banda casi ilimitado). Posteriormente se comprobó la conectividad enviando mensajes y alarmas (véase Ilustración 15).

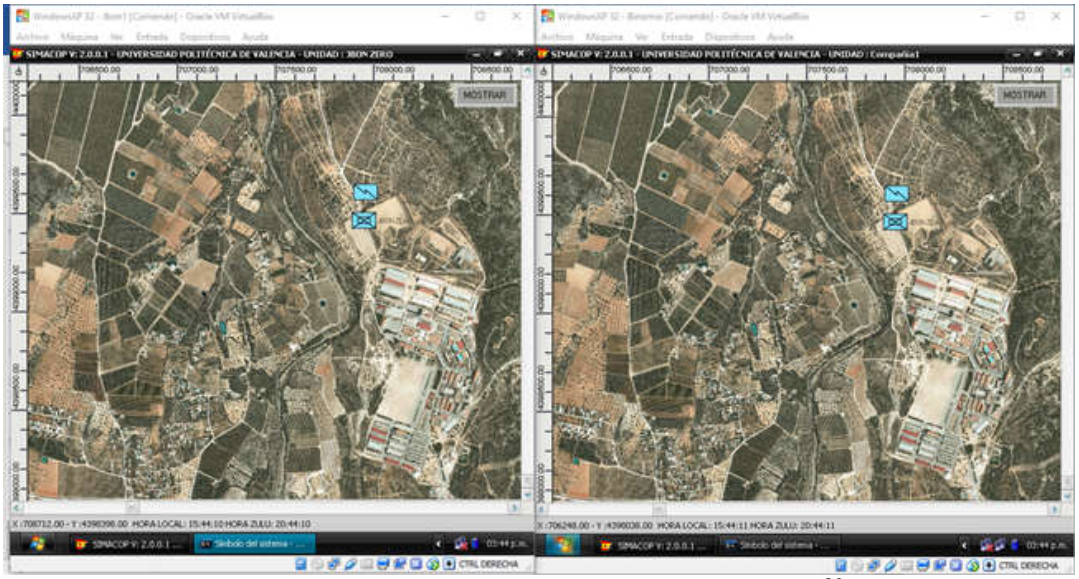



**Ilustración 15 - Pruebas de conectividad FFT.**[30]

Los resultados obtenidos en este entorno fueron los siguientes:

[30] Fuente: Elaboración propia.

| Envío de 3 mensajes de JBON a COMPAÑÍA. | |
|---|---|
| MENSAJE | TIEMPO EN LLEGAR |
| 1 | 7 s |
| 2 | 36 s. |
| 3 | 11s |

| Envío de 3 mensajes de JBON a COMPAÑÍA. | |
|---|---|
| MENSAJE | TIEMPO EN LLEGAR |
| 1 | 31 s |
| 2 | 6s |
| 3 | 10s |

**Tabla 2 - Envío de mensajes.**

Se puede observar que a pesar de que estamos trabajando en condiciones ideales (el ping entre los dos equipos es inferior a 10ms), el sistema FFT tarda mucho en enviar los mensajes, replicar las amenazas o actualizar la posición. Una explicación de esta lentitud podría ser que el sistema está diseñado para no saturar la red (el ancho de banda disponible en una radio PR4G es inferior a 10Kbps) y para ello genera muy poco tráfico. Una funcionalidad que se debería incorporar a FFT es que adapte la velocidad de refresco al ancho de banda disponible para disminuir la latencia. También se ha comprobado que el programa actualiza la posición GPS cada 10 segundos en el ordenador que está conectado directamente con el receptor GPS, esto hace que en el mejor de los casos y en condiciones ideales la posición se replicaría en 30 segundos. No obstante, en estas mismas maniobras se comprobó que en el sistema FFT las posiciones tardaban en actualizarse de 1 a 5 minutos, dependiendo de la distancia, y usando como medios de trasmisión las radios PR4G (VHF), HARRIS 5800 (HF) o HARRIS G117 (Satélite).

## 4.5 Pruebas de conectividad en el FFT usando Arduino.

La siguiente prueba fue integrar la posición que enviaba el Arduino remoto en el FFT. Esta prueba se realizó en las instalaciones del Regimiento de Transmisiones 21 de Marines (Valencia) y se dispuso la misma arquitectura que en el caso anterior. Además, virtualmente se habilitó el puerto COM3, que es el que usa el Arduino para conectarse con el portátil, para que una máquina virtual con Windows XP (Compañía) pudiera acceder a la posición. mientras que en la otra máquina virtual (JBON ZERO) se simuló el movimiento usando el programa "***sim_marines-1.3.exe***"[31] Así se comprobó que ambas posiciones se actualizaban. Los resultados fueron satisfactorios, la Compañía varió su posición conforme se movía el dispositivo, (véase Ilustración 16).

[31] Este programa pertenece al FFT y se usa para simular posiciones de unidades en el mapa sin tener que obtener la posición a través del GPS, se usa solo para pruebas.

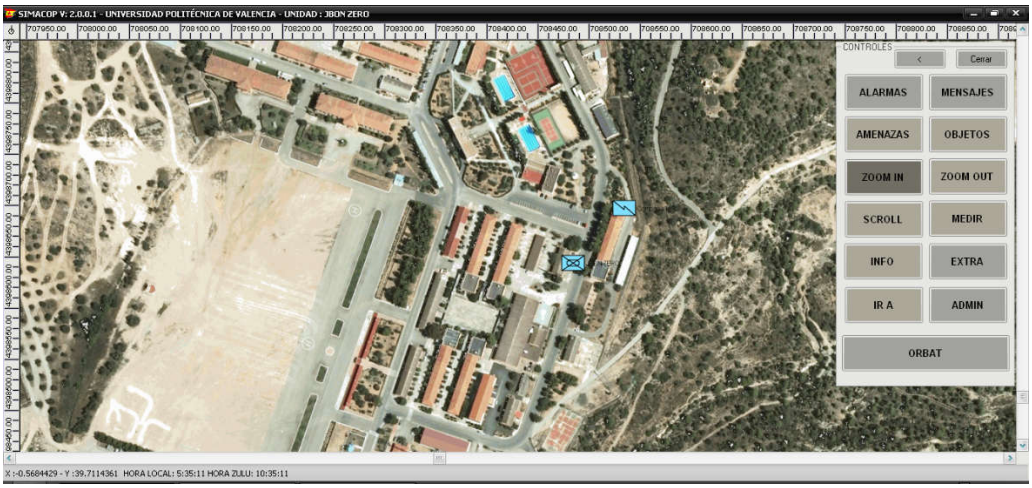



**Ilustración 16 - FFT recibiendo posición desde Arduino.**[32]

## 4.6 Prueba de autonomía

En esta prueba se comprobó el consumo eléctrico del dispositivo remoto. La estimación de la autonomía del dispositivo se realizó empleando una batería de 2200mAh como fuente de energía.

Para realizar la estimación, se encendió el dispositivo incluyendo la transmisión de los datos y la pantalla LCD y se midió los mA que consumía el dispositivo para funcionar a pleno rendimiento. Dicha medición se realizó conectando un polímetro en serie entre la batería y el Arduino.

Tras realizar las pruebas, el polímetro dio un valor de 200mA de consumo instantáneo, por lo que se podría estimar que la autonomía del dispositivo es de 2200mA/200 = 11 horas. No obstante, puesto que ninguna maquina es 100% eficiente y la batería no va a proporcionar toda su carga al dispositivo, es recomendable ser más conservadores y afirmar que el dispositivo puede estar al menos 8 horas encendido.

## 4.7 Prueba con transmisión de posición por P4RG

En este apartado se describe una prueba realizada en el Regimiento de Transmisiones 21 de Marines (Valencia) durante los días 18 y 19 de octubre de 2016. La prueba consistió en transmitir a través de la radio PR4G la posición remota obtenida desde el Arduino en movimiento a otro vehículo. Este último vehículo mostró en el FFT las nuevas posiciones en tiempo real, tanto del Arduino en movimiento como la suya propia, (véase Ilustración 17).

[32] Fuente: Elaboración propia.


**Ilustración 17 - Esquema prueba de posición con radio PR4G y Arduino.**[33]

La prueba finalmente fue satisfactoria, replicándose las posiciones del Arduino en movimiento en el vehículo 2 en menos de un minuto. Se ha de recalcar que los vehículos estaban a menos de 20 metros. El Anexo F contiene más documentación gráfica de cómo fue realizado la prueba.

[33] Fuente: Elaboración propia.

# Capítulo 5. Conclusiones

## 5.1 Principales conclusiones

En este trabajo se ha demostrado que, usando tecnología libre basada en Arduino, se puede diseñar un dispositivo fácilmente transportable por un soldado para recoger una serie de datos de interés (como por ejemplo la posición). Estos datos posteriormente se transmiten a un nodo central que puede integrarlos en el sistema FFT.

El funcionamiento de la solución propuesta se ha validado en pruebas en interiores y exteriores, obteniéndose unos alcances de en torno a 150 metros en exteriores y de 40 metros en interiores. La autonomía del dispositivo es superior a 8 horas, por lo que es adecuada dada la duración de las misiones a realizar por los soldados. Finalmente, se ha conseguido integrar la posición de los soldados en el sistema FFT.

Los problemas encontrados a la hora de realizar este proyecto han sido principalmente de tipo técnico. Debido a la dificultar de integrar componentes electrónicos entre sí, se ha tenido que valorar la compatibilidad y el consumo de los módulos. Además, debido a que no existe ningún otro producto igual en el mercado, se ha tenido que realizar la programación prácticamente desde cero, partiendo de unos ejemplos. Otro problema fundamental ha sido la integración en el FFT. En este caso, lo ideal sería poder modificar el programa FFT directamente para que aceptase los valores de los sensores. Como el FFT solo muestra la posición de la unidad y su código fuente no está disponible, únicamente ha sido posible incluir la posición de los soldados en el mismo. Para ello, ha sido necesario desarrollar un mecanismo específico que permitiera incluir estos datos de posición en el FFT.

En resumen, podemos concluir que el sistema ha funcionado satisfactoriamente y que se han cumplido los objetivos iniciales propuestos.

## 5.2 Líneas futuras de trabajo

A continuación, se detallan las posibles mejoras que se deberían incorporar al sistema propuesto para su uso en un entorno real. Para ello se ha empleado la herramienta casa de la calidad[34], que ha dado como resultado que el dispositivo debería ser mejorado en los aspectos de seguridad, alcance y materiales.

En primer lugar, sería obligatorio mejorar la calidad de los materiales empleados. Es necesario reducir el tamaño y el peso, no siendo muy complicado dada la tecnología existente actualmente. Además, el sistema debe hacerse estanco y resistente a las inclemencias del tiempo. También se debería ajustar la luminosidad de los LED integrados para poder usar el dispositivo en ambientes nocturnos. Respecto a la seguridad, sería necesario emplear algún mecanismo de cifrado en la información transmitida.

Por otro lado, para mejorar el alcance se debería utilizar otro tipo de módulos de comunicación que trabajen a frecuencias más bajas. Este cambio es posible ya que

[34] Ver Anexo C – Estudio de mercado.

la cantidad de datos a transmitir no es elevada[35]. Además, la autonomía del dispositivo no se ve afectada con este cambio. Una propuesta seria usar módulos de transmisión para Arduino que funcionan a la frecuencia de 433 MHz. Otra estrategia para mejorar el alcance sería emplear una antena de mayor ganancia encima del vehículo en el Arduino local y varias antenas en el Arduino remoto.

Respecto a los problemas encontrados en la integración con el sistema FFT, estos se tendrían solucionar cambiando la programación del sistema. Por ejemplo, se podría emplear aceleración por hardware de la tarjeta gráfica o adaptar la velocidad de envío de información GPS dependiendo de la velocidad disponible. También sería de utilidad disponer de una versión compatible con Windows 10. Además, si se quiere introducir los datos de los sensores, se debería modificar el sistema FFT para que aceptara y mostrara estos datos.

La latencia en el programa se podría disminuir filtrando el contenido no necesario de información GPS que proviene de las radios y transmitir solo la sentencia NMEA $GPRMC, que es la que usa el FFT. La realidad es que las radios transmiten mayor cantidad de información que al final se desecha.

De cara al futuro, estos sensores se podrían fácilmente integrar con la radio Spearnet[36] y ser transmitidos a mayor distancia a través de esta. Se podrían añadir otro tipo de sensores, como detectores de sustancias químicas en el aire o de humo e incorporar esta información a los vehículos. También se podría emplear sensores LIDAR[37], que es una tecnología que permite valorar distancias usando un emisor laser. Con esto se podría crear una especie de burbuja alrededor del vehículo ya que se podría conocer todo lo que hay en su entorno.

## 5.3 Comentarios personales

Este Trabajo Fin de Grado representa el fin de mis estudios en el Centro Universitario de la Defensa, pero no representa el fin de estudiar. Gracias a este trabajo he descubierto un nuevo camino que quiero seguir en mi formación, me gustaría profundizar en el tema de los automatismos y módulos de control con sensores. Además, utilizando los conocimientos que tenía sobre programación adquiridos durante el grado, he conseguido hacer funcionar un sistema relativamente complejo. También he obtenido conocimientos sobre electrónica que me serán útiles en mi futuro profesional.

[35] En cualquier sistema de comunicaciones inalámbrico, cuanto más baja es la frecuencia de transmisión mayor es el alcance y menor el ancho de banda disponible.
[36] Consultar Anexo C para más información de esta radio.
[37] Laser Imaging Detection and Ranging.

# Anexos

## Anexo A – Presupuesto

En este apartado se mostrará el coste real de los componentes utilizados para montar los prototipos.

| COMPONENTE | COSTE (€) |
|---|---|
| Arduino UNO rev3 | 22,33 |
| Arduino Mega 2560 R3 | 22,40 |
| Módulo de sensor de pulso del ritmo cardíaco | 5,36 |
| Módulo Ublox NEO - 6M GPS | 11,99 |
| Módulo NRF24L01+ LNA SMA Antena 2.4G X 2 | 18,75 |
| Termistor LM35 | 1,00 |
| Fotoresistor GL5528 | 0,30 |
| Pantalla LCD X 2 | 8,70 |
| Batería externa 2200mAh | 7,99 |
| Cableado | 5,99 |
| Cajas herméticas de plástico | 3,00 |
| Portes | 6,00 |
| **TOTAL** | 113,81€ |

**Tabla 3 - Presupuesto.**

Todos estos precios corresponden a una compra realizada el día 15 de junio de 2016 al portal Amazon. El coste de estos componentes podría disminuir al comprarlos directamente al fabricante.

En esta memoria se considera que un único dispositivo móvil se conecta a una estación central. Si tuviéramos más dispositivos móviles y una sola estación central el coste por unidad también disminuiría.

## Anexo B- Presentación del prototipo.

### Esquema de conexionado.

En el siguiente esquema realizado con el programa Fritzing[38] se muestra como se han realizado las conexiones entre los Arduinos y los diferentes módulos/sensores.

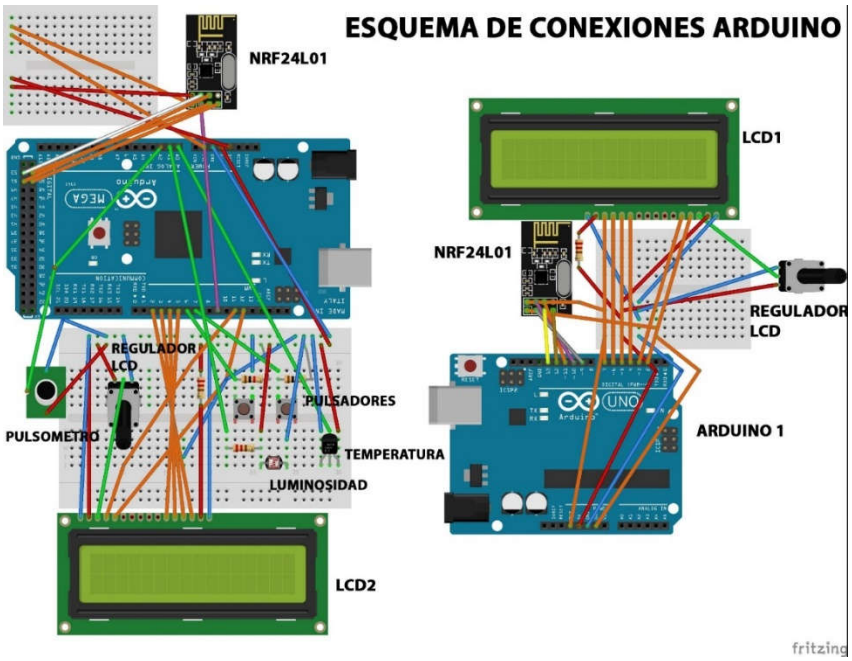



**Ilustración 18 - Esquema de conexiones en Arduino.[39]**

### Prototipo construido.

En este apartado se muestran dos fotografías del prototipo construido para demostrar que el sistema funciona. Se pueden apreciar los diferentes elementos como el LCD, los botones, la antena de transmisión, el receptor GPS, y los sensores.

[38] Programa gratuito descargable desde http://fritzing.org/home/
[39] Fuente: Elaboración propia.

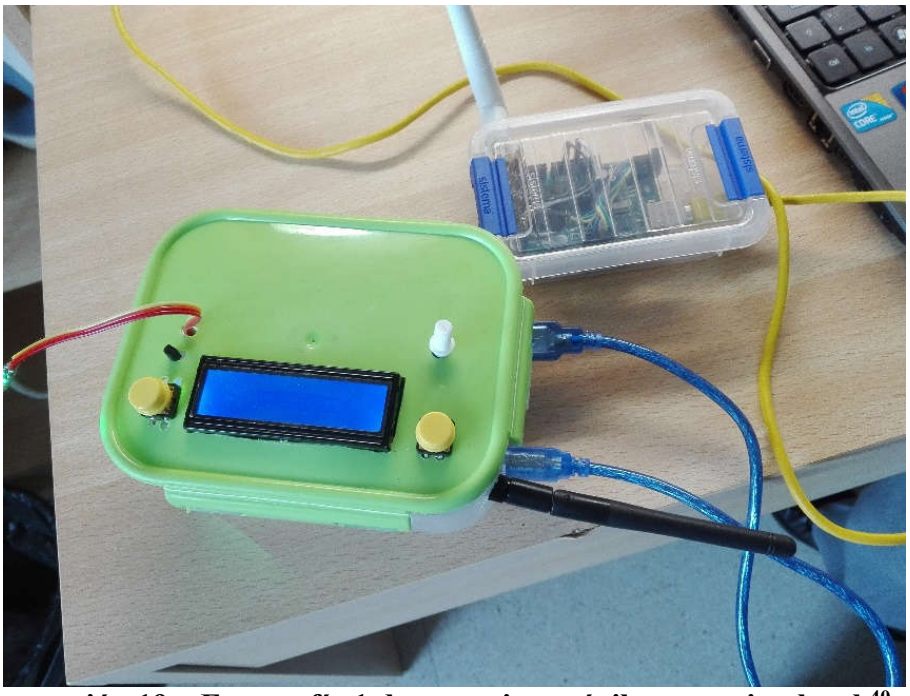


**Ilustración 19 – Fotografía 1 de prototipo móvil y prototipo local.**[40]

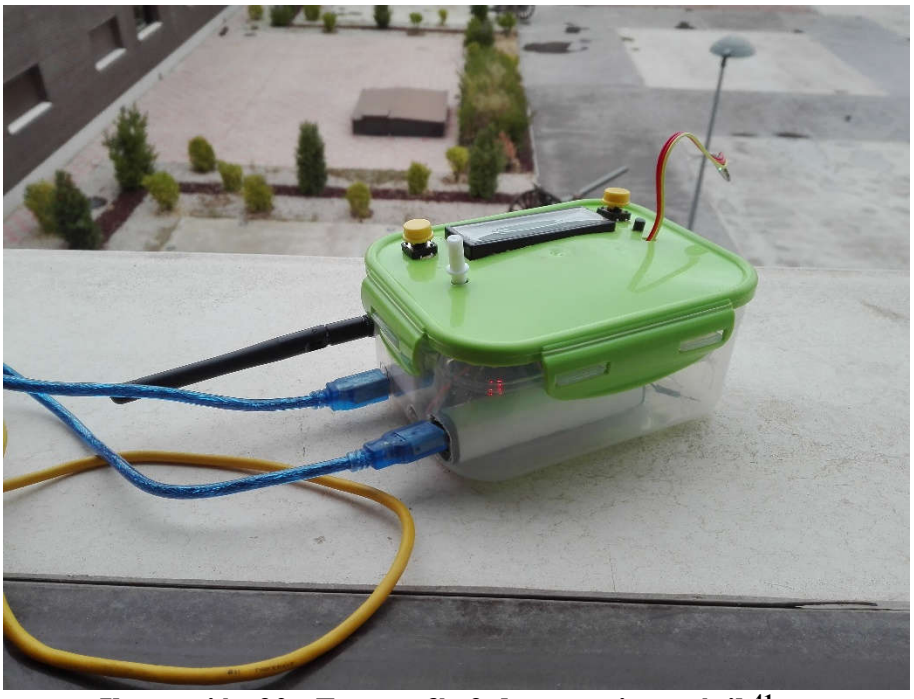


**Ilustración 20 - Fotografía 2 de prototipo móvil.**[41]

[40] Fuente: Elaboración propia.
[41] Fuente: Elaboración propia.

## Anexo C – Estudio de mercado

Antes de diseñar un nuevo dispositivo se deberá analizar las tendencias de mercado en base a los productos existentes. Aunque existen infinidad de dispositivos dedicados a la comunicación dentro del Ejército de Tierra, se analizarán aquí los más utilizados que sean también portátiles. A pesar de que no exista en el mercado ningún dispositivo igual al presentado en esta memoria, este análisis dará una idea de lo que el mercado demanda. Para ello se han seleccionado tres: la PR4G V3, la PNR 500 y la Spearnet.

**Ilustración 21 - PR4G v3.**[42]

**Ilustración 22 - PNR500.**[43]

**Ilustración 23 – SPEARNET.**[44]

A continuación, se expondrá una comparativa con las características más importantes de cada radio. Estas características se han seleccionado según los criterios de importancia expuestos por el personal del RT21 que las emplean.

| | PR4G v3 | PNR500 | SPEARNET |
|---|---|---|---|
| Rango de frecuencias | 30-88 MHz - VHF | 30 MHz a 2 GHz - UHF | 30 MHz a 2 GHz - UHF |
| Medidas de protección | SI | NO | SI |
| Alcance | 8 km | 500 metros | 2000 metros |
| Peso | 7000g | 450g | 700g |
| Autonomía | 24 horas | 8 horas | 12 horas |
| Transmisión de video | NO | NO | SI |
| Capacidad mesh[45] | NO | NO | SI |
| Posicionamiento GPS | SI | NO | SI |

**Tabla 4 - Comparativa radios**

Todos estos valores son teóricos y obtenidos de las características proporcionadas del fabricante y publicados en la página web del Ejército de Tierra. Es obvio decir que dependiendo del terreno donde estemos probando la radio el alcance será menor. Este alcance disminuye notablemente en zonas urbanas cuando se emplean en altas

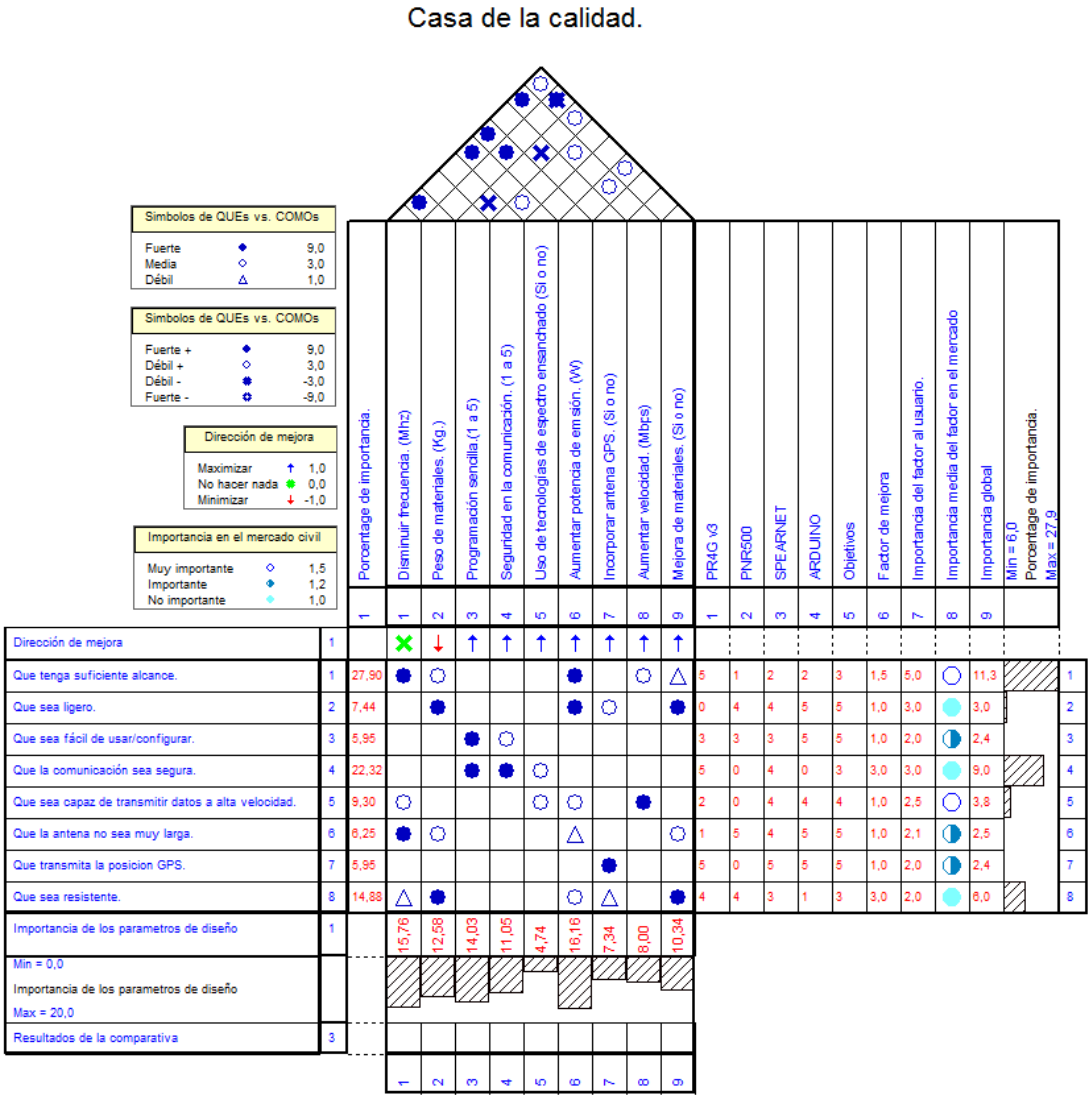
[42] Fuente: Elaboración propia.
[43] Fuente: Elaboración propia.
[44] Fuente: Elaboración propia.
[45] Tecnología mesh: es una red inalámbrica mallada en la que cada dispositivo actúa de repetidor automáticamente, para permitir establecer enlace con dispositivos que no están al alcance del dispositivo original.

frecuencias debido a la atenuación que sufre la señal al atravesar las paredes de los edificios. Por ejemplo, se ha comprobado experimentalmente que la SPEARNET y la PNR500 pierden el enlace si se posiciona un vehículo de dimensiones medias entre la radio emisora y la receptora. Esto implica que únicamente funcionan bien cuando tienen visión directa. Por el contrario, la PR4G v3 no presenta estos problemas, pero al trabajar en frecuencias más bajas debe emitir a más potencia, por lo que el peso del equipo se incrementa. Además, el posicionamiento GPS solo se podrá utilizar en terreno abierto, siendo imposible con la tecnología existente por el momento, su uso en interiores.

Aparte de las características citadas anteriormente, también se preguntó a los operadores de este tipo de equipos en el RT21 sus preferencias de uso y gustos, así como las mejoras que consideran que se deberían añadir a los mismos. Las respuestas más repetidas son las siguientes:


Para establecer una relación entre lo que demanda el usuario y lo que nuestro producto debe ofrecer se realizará una casa de la calidad. El primer paso ya se ha realizado y es establecer las necesidades más importantes que demandan los usuarios. Una vez se han seleccionado los "*Qué*" se procederá a establecer los "*Cómo*", lo que implica establecer el modo técnico de adaptar la tecnología existente a las demandas de los usuarios.

Finalmente se presenta la casa de la calidad realizada con el programa QFD Capture, en la cual se pueden observar aspectos a mejorar:




Antes de analizar los datos expuestos en la casa de la calidad se debe tener en cuenta que nuestro dispositivo Arduino no posee ningún tipo de seguridad en la comunicación. La casa de la calidad nos muestra que para cumplir con las demandas de los usuarios en primer lugar se debe aumentar la potencia de emisión, en segundo lugar, se debería utilizar una frecuencia menor, y por último se debería usar una programación sencilla. En un futuro si el proyecto parece viable se debería trabajar en esos aspectos, aumentando la potencia de emisión o cambiando la antena por una de más ganancia. Además, sería necesario diseñar un protocolo de comunicación seguro y fiable en frecuencias más bajas, usar programación intuitiva y diseñar una carcasa más resistente pero ligera.

## Anexo D – Codificación NMEA

NMEA (*National Marine Electronic Asociation*) es un protocolo usado para la navegación tanto marítima como terrestre. En un principio el NMEA se creó para ser utilizado en la navegación marítima, así el primero en aparecer fue el NMEA 180 simple, que permitía transmitir datos a un piloto automático del desvío de desviación a babor o estribor respecto de una trayectoria predeterminada. Una vez fue avanzando la técnica, ya se transmitían diferentes datos de velocidad del barco, dirección de viento, profundidad, fecha y hora, nº de *waypoint* y coordenadas. A una velocidad de 4.800 Baudios.

Actualmente NMEA se ha estandarizado a nivel mundial. En cada sentencia además de la posición se envía información de posición de satélites, que satélites se están recibiendo, su posición, el *datum*, la aceleración, la altura [3]. Dependiendo del comienzo de la sentencia contendrá unos datos u otros. Aquí un ejemplo de sentencia:

$GPGSV,1,1,13,02,02,213,,03,-3,000,,11,00,121,,14,13,172,05*67

En este caso $GPGSV nos indica GPS *Satellites in view* donde:

= Número total de mensajes de este tipo en el ciclo.

= Numero de mensaje.

= Numero de satélites a la vista

= Numero de satélite

= Elevación en grados, 90º máximo.

= Azimut, desde el polo norte, 000 a 359

= SNR, 00-99 dB

8-11 = Información del segundo satélite, misma información que 4-7

12-15 = Información del tercer satélite, misma información que 4-7

16-19 = Información del cuarto satélite, misma información que 4-7

En el trabajo se ha utilizado el formato $GPRMC (*Recommended minimum specific GPS/Transit data*), puesto que es el único que aceptaba el FFT, un ejemplo es:

$GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68

| | |
|---|---|
| **225446** | Hora 22:54:46 UTC. |
| **A** | A = OK, V = Problemas en recepción. |
| **4916.45,N** | Latitud 49 grados. 16.45 min Norte. |
| **12311.12,W** | Longitud 123 grados. 11.12 min Este. |
| **000.5** | Velocidad en tierra en nudos. |
| **054.7** | Recorrido realizado. |
| **191194** | Fecha 19 Noviembre 1994. |
| **020.3,E** | Variación magnética 20.3º Este. |

| | |
|---|---|
| ***68** | Suma de comprobación. |

En el prototipo se ha utilizado un receptor Ublox NEO - 6M GPS, el cual aparte del formato $GPRMC ofrece más información, pero por no ser aplicable al proyecto se descarta. Este receptor GPS se ha comprobado que tarda unos 40s en establecer una posición valida cuando se enciende “en frio”, esto significa cuando no tiene una posición guardada anteriormente. Cuando se enciende “en caliente” (significa que ya se habían tomado datos de posición de la zona y están guardadas en memoria), en apenas 10 segundos ya transmite correctamente la posición.

## Anexo E – Especificaciones del módulo de comunicaciones nRF24L01.

Estas especificaciones han sido extraídas de la hoja técnica[46] del fabricante Nordic Semiconductor.

### Características del nRF24L01:

- Radio

Banda de 2.4Ghz de uso global.
126 canales de transmisión.
Modulación GFSK.
Velocidad de 1 a 2Mbps

- Transmisor

Potencia de salida programable: 0, -6, -12 o -18dBm
Consumo: 11.3mA at 0dBm

- Receptor

Filtros integrados de canal
Consumo: 12.3mA at 2Mbps
Sensibilidad: -82dBm a 2Mbps.
Sensibilidad -85dBm a 1Mbps.
Ganancia LNA programable.

- Sintetizador

Sintetizador integrado
Filtro de bucles y varactor VCO

- Enhanced ShockBurst™

Gestión de longitud de paquete dinámica 1 a 32 bytes.
Gestión automática de paquetes
6 canales de datos MultiCeiver™ para redes en estrella.

- Gestión energética:

Regulador integrado de voltaje.
Rango: 1.9 a 3.6V
Modos de ahorro energético.
22uA en modo en espera, 900nA en modo apagado.

### Diagrama de bloques:

[46] La hoja técnica del producto NRF24L01 se puede consultar en: https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf (Consultado el 20 de Septiembre de 2016).

Anexos.

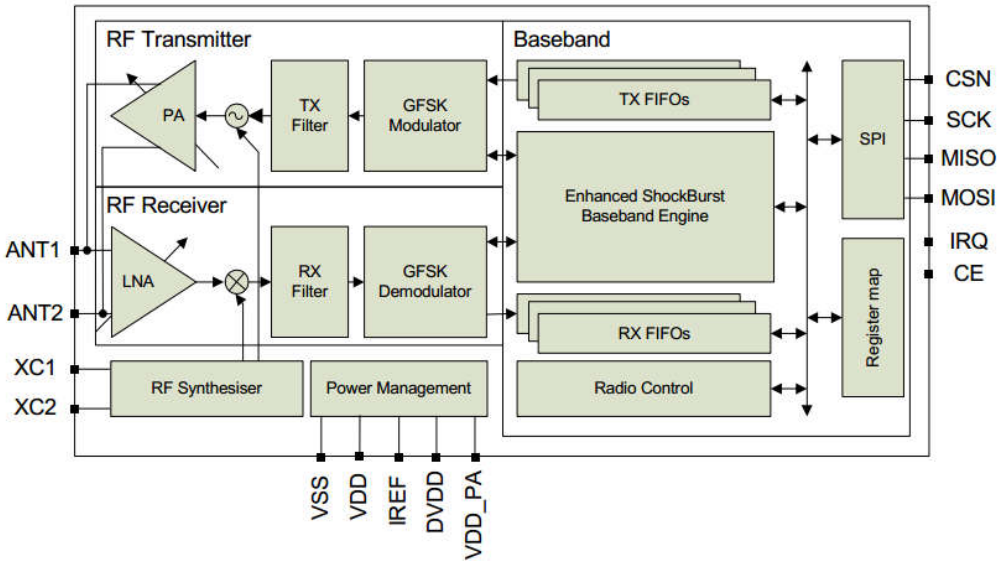



**Ilustración 24 - Diagrama de bloques del nRF24L01.**[47]

[47] Fuente: Hoja técnica nRF24L01

X

## Anexo F – Fotografías realizadas.

- **Pruebas de distancia.**

**Ilustración 25 - Preparación de pruebas de alcance.**

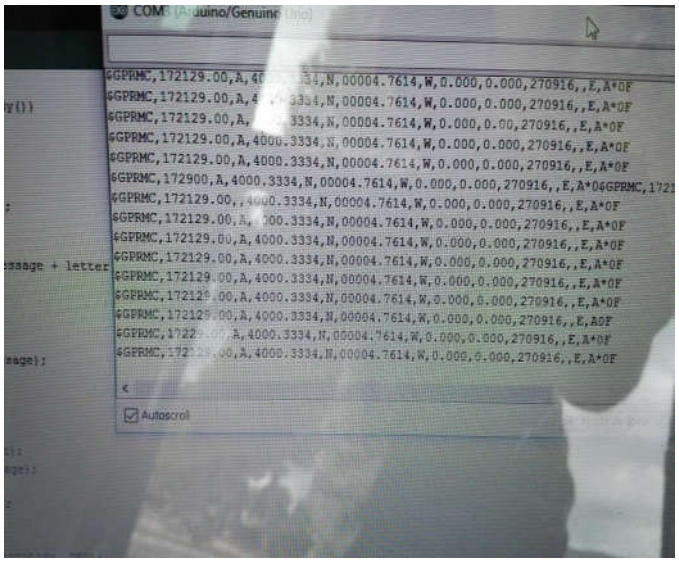


**Ilustración 26 - Recibiendo posición GPS del Arduino remoto.**

- **Pruebas Arduino y FFT.**

**Ilustración 27 - Arduino en el Mercurio IP**

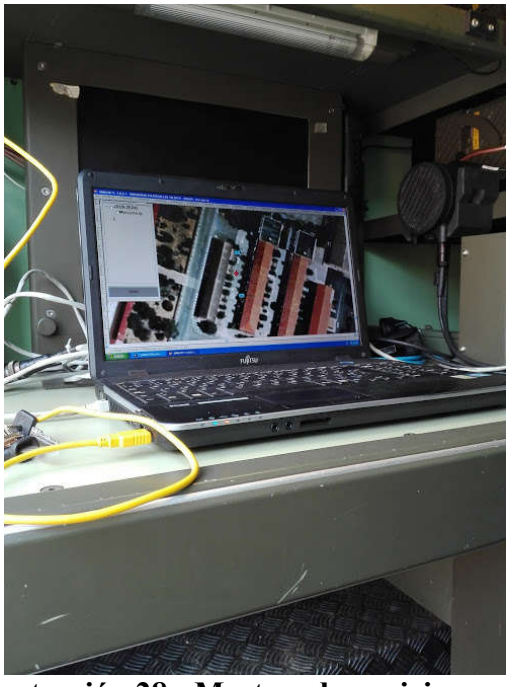


**Ilustración 28 - Mostrando posiciones en el FFT**

## Anexo G – Código de programación.

En este apartado se muestra el código de programación del Arduino remoto.

Anexos.

Se han utilizado las librerías SPI, MIRF,nRF24L01, LiquidCrystal y parte de código extraído del manual de uso del sensor de pulso (pulsesensor) y del código de TinyGPS++.

```
//Arduino remoto MEGA
//Incluir las librerías necesarias.
#include <SPI.h>
#include <Mirf.h>
#include <nRF24L01.h>
#include <MirfHardwareSpiDriver.h>

//Parte RADIO
int ratio = 0;
int val = 0;
int datorecibido;

//Parte LCD
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
//Pines y variables switches
const int switch1Pin = 6;
int switch1State = 0;
int prevSwitch1State = 0;
const int switch2Pin = 7;
int switch2State = 0;
int prevSwitch2State = 0;
int PosMenu = 0;
const int PosMenuMax = 1;
const int PosMenuMin = 1;

//Pines Sensores
int SensorTemp = 0 ;
int SensorLuz = 1 ;
int SensorPulso = 2 ;

//Parte Sensor Pulso
//  Variables
int pulsePin = 2;
int blinkPin = 13;
int fadePin = 5;
int fadeRate = 0;
volatile int BPM;
volatile int Signal;
volatile int IBI = 600;
volatile boolean Pulse = false;
volatile boolean QS = false;

//Parte GPS
#define powerpin 4
#define GPSRATE 9600
#define BYTE 1
#define BUFFSIZ 90
char buffer[BUFFSIZ];
char *parseptr;
char buffidx;
uint8_t hour, minute, second, year,
month, date;
uint32_t latitude, longitude;
uint8_t groundspeed, trackangle;
char latdir, longdir;
char status;

void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.print("Soldado 2.0");

  //Parte RADIO
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);
  Mirf.cePin = 9;
  Mirf.csnPin = 10;
  Mirf.spi = &MirfHardwareSpi;
  Mirf.init();
  Mirf.setRADDR((byte
*)"ArduMega");
  Mirf.setTADDR((byte
*)"ArduUNO");
  Mirf.payload = sizeof(byte);
  Mirf.channel = 102;
  Mirf.config();
  Mirf.configRegister(RF_SETUP,
0x0e);
  Mirf.configRegister(EN_AA, 0x00);
  Mirf.configRegister(SETUP_RETR,
0x05);
  //Parte Sensor Pulso
  pinMode(blinkPin, OUTPUT);
  pinMode(fadePin, OUTPUT);
  interruptSetup();

  //Parte GPS
  if (powerpin) {
    pinMode(powerpin, OUTPUT);
  }
  pinMode(13, OUTPUT);
  Serial1.begin(GPSRATE);
  digitalWrite(powerpin, LOW);
}

void transmit(const char *string) {
  byte c;
```

```
  for (int i = 0; string[i] != 0x00; i++) {
    c = string[i];
    Mirf.send(&c);
    delay(80);
    while (Mirf.isSending() );
  }
}
uint32_t parsedecimal(char *str) {
  uint32_t d = 0;

  while (str[0] != 0) {
    if ((str[0] > '9') || (str[0] < '0'))
      return d;
    d *= 10;
    d += str[0] - '0';
    str++;
  }
  return d;
}

void readline(void) {
  char c;
  buffidx = 0;
  while (1) {
    c = Serial1.read();
    if (c == -1)
      continue;
    if (c == '\n')
      continue;
    if ((buffidx == BUFFSIZ - 1) || (c == '\r')) {
      buffer[buffidx] = 0;
      return;
    }
    buffer[buffidx++] = c;
  }
}


void loop() {
  //Parte GPS
  uint32_t tmp;
  readline();
  if (strncmp(buffer, "$GPRMC", 6) == 0) {
    // hhmmss time data
    parseptr = buffer + 7;
    tmp = parsedecimal(parseptr);
    hour = tmp / 10000;
    minute = (tmp / 100) % 100;
    second = tmp % 100;
    parseptr = strchr(parseptr, ',') + 1;
    status = parseptr[0];
    parseptr += 2;
    // grab latitude & long data
    // latitude
    latitude = parsedecimal(parseptr);
    if (latitude != 0) {
      latitude *= 10000;
      parseptr = strchr(parseptr, '.') + 1;
      latitude += parsedecimal(parseptr);
    }
    parseptr = strchr(parseptr, ',') + 1;
    // read latitude N/S data
    if (parseptr[0] != ',') {
      latdir = parseptr[0];
    }
    // longitude
    parseptr = strchr(parseptr, ',') + 1;
    longitude = parsedecimal(parseptr);
    if (longitude != 0) {
      longitude *= 10000;
      parseptr = strchr(parseptr, '.') + 1;
      longitude += parsedecimal(parseptr);
    }
    parseptr = strchr(parseptr, ',') + 1;
    // read longitude E/W data
    if (parseptr[0] != ',') {
      longdir = parseptr[0];
    }
    // groundspeed
    parseptr = strchr(parseptr, ',') + 1;
    groundspeed = parsedecimal(parseptr);
    // track angle
    parseptr = strchr(parseptr, ',') + 1;
    trackangle = parsedecimal(parseptr);
    // date
    parseptr = strchr(parseptr, ',') + 1;
    tmp = parsedecimal(parseptr);
    date = tmp / 10000;
    month = (tmp / 100) % 100;
    year = tmp % 100;
    if (strncmp(buffer, "$GPRMC", 6) == 0) {
      transmit(buffer);
      transmit("~");
      Serial.println(buffer);
    }
  }
  //SENSORES
  //Lectura sensor temperatura
  delay(200);
  int lecturaTemp = analogRead(SensorTemp);
  float voltaje = 5.0 / 1024 * lecturaTemp;
```

```
      float temp = voltaje * 100 - 50 ;
      Serial.print("La temperatura es: ");
      Serial.println(temp) ;

      //Lectura sensor luz
      int lecturaLuz = analogRead(SensorLuz);
      Serial.print("Luminosidad: ");
      if (lecturaLuz < 50) {
        Serial.println("+++");
      }
      else if (lecturaLuz > 50 and lecturaLuz < 100) {
        Serial.println("++-");
      }
      else if (lecturaLuz > 100 and lecturaLuz < 200) {
        Serial.println("+--");
      }
      else if (lecturaLuz > 201) {
        Serial.println("---");
      }

      //Parte SensorPulso
      if (QS == true) {
        fadeRate = 255;
        serialOutputWhenBeatHappens();
        QS = false;
      }
      ledFadeToBeat();
      delay(500);

      //Parte LCD
      switch1State = digitalRead(switch1Pin);
      switch2State = digitalRead(switch2Pin);
      delay(70);
      lcd.setCursor(0, 0);
      lcd.print("Constantes: ");
      Serial.println(PosMenu);

      //Imprimir en pantalla posición que estemos en el menu.
      lcd.setCursor(0, 1);
      switch (PosMenu) {
        case 0:
          lcd.print("Temp: ");
          lcd.setCursor(7, 1);
          lcd.print(temp);
          break;
        case 1:
          lcd.print("Luz: ");
          lcd.setCursor(5, 1);
          if (lecturaLuz < 50) {
            lcd.print("+++");
          }
          else if (lecturaLuz > 50 and lecturaLuz < 100) {
            lcd.print("++-");
          }
          else if (lecturaLuz > 100 and lecturaLuz < 200) {
            lcd.print("+--");
          }
          else if (lecturaLuz > 201) {
            lcd.print("---");
          }
          break;
        case 2:
          lcd.print("BPM: ");
          lcd.setCursor(8, 1);
          lcd.print(BPM);
          break;
      }

      //Cambio de menú dependiendo del botón izq. o dcho. que pulsemos.
      if (prevSwitch1State != switch1State or prevSwitch2State != switch2State) {
        if (switch1State == LOW and PosMenuMax >= PosMenu) {
          PosMenu++;
          prevSwitch1State = switch1State;
        }
        if (switch2State == LOW and PosMenuMin <= PosMenu) {
          PosMenu--;
          prevSwitch2State = switch2State;
        }
      }
      //Parte radio - Recibir
      Mirf.getData((byte *) &datorecibido);
      Serial.println(datorecibido);
    }

    //INO EXTERNO CON LA PARTE DE LA INTERRUPCION CUANDO SE DETECTA UN PULSO
    volatile int rate[10];
    volatile unsigned long sampleCounter = 0;
    volatile unsigned long lastBeatTime = 0;
```

```
volatile int P =512;
volatile int T = 512;
volatile int thresh = 525;
volatile int amp = 100;
volatile boolean firstBeat = true;
volatile boolean secondBeat = false;


void interruptSetup(){
  TCCR2A = 0x02;
  TCCR2B = 0x06;
  OCR2A = 0X7C;
  TIMSK2 = 0x02;
  sei();
}

ISR(TIMER2_COMPA_vect){
  cli();
  Signal = analogRead(pulsePin);
  sampleCounter += 2;
  int N = sampleCounter - lastBeatTime;
  if(Signal < thresh && N > (IBI/5)*3){
    if (Signal < T){
      T = Signal;
    }
  }
  if(Signal > thresh && Signal > P){
    P = Signal;
  }
  if (N > 250){
    if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) ){
      Pulse = true;
      digitalWrite(blinkPin,HIGH);
      IBI = sampleCounter - lastBeatTime;
      lastBeatTime = sampleCounter;

      if(secondBeat){
        secondBeat = false;
        for(int i=0; i<=9; i++){
          rate[i] = IBI;
        }
      }

      if(firstBeat){
        firstBeat = false;
        secondBeat = true;
        sei();
        return;
      }
      word runningTotal = 0;
      for(int i=0; i<=8; i++){
        rate[i] = rate[i+1];
        runningTotal += rate[i];
      }

      rate[9] = IBI;
      runningTotal += rate[9];
      runningTotal /= 10;
      BPM = 60000/runningTotal;
      QS = true;
    }
  }

  if (Signal < thresh && Pulse == true){
    digitalWrite(blinkPin,LOW);
    Pulse = false;
    amp = P - T;
    thresh = amp/2 + T;
    P = thresh;
    T = thresh;
  }

  if (N > 2500){
    thresh = 512;
    P = 512;
    T = 512;
    lastBeatTime = sampleCounter;
    firstBeat = true;
    secondBeat = false;
  }

  sei();
}

//INO EXTERNO CON FUNCIONES PARA EL PULSO
void ledFadeToBeat(){
  fadeRate -= 15;
  fadeRate = constrain(fadeRate,0,255);
  analogWrite(fadePin,fadeRate);
}

void serialOutputWhenBeatHappens(){
  Serial.print("*** Pulso encontrado *** ");
  Serial.print("BPM: ");
  Serial.print(BPM);
  Serial.println("  ");
}
```

Anexos.

Aquí se muestra el código usado en el Arduino local.

```
//Arduino UNO
//Incluir las librerías necesarias.
#include <SPI.h>
#include <Mirf.h>
#include <nRF24L01.h>
#include <MirfHardwareSpiDriver.h>
#include <LiquidCrystal.h>

//Parte Radio
int datorecibido;
int datoenviado = 123;
String message;
//Parte LCD
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);
const int switch1Pin=0;
int switch1State=0;
int prevSwitch1State=0;
const int switch2Pin=1;
int switch2State=0;
int prevSwitch2State=0;
int PosMenu=0;
const int PosMenuMax=1;
const int PosMenuMin=1;

void setup() {
lcd.begin(16,2);
lcd.print("Soldado 2.0");

  //Parte radio
  Serial.begin(9600);
  Mirf.cePin = 9;
  Mirf.csnPin = 10;
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);
  Mirf.spi = &MirfHardwareSpi;
  Mirf.init();
  Mirf.setRADDR((byte
*)"ArduUNO");
  Mirf.setTADDR((byte
*)"ArduMega");
  Mirf.payload = sizeof(byte);
  Mirf.channel = 102;
  Mirf.config();
  Mirf.configRegister(RF_SETUP,
0x0e);
  Mirf.configRegister(EN_AA, 0x00);
  Mirf.configRegister(SETUP_RETR,
0x05);
  // Read and print RF_SETUP
  byte rf_setup = 0;
  Mirf.readRegister(        RF_SETUP,
&rf_setup, sizeof(rf_setup) );
  Serial.print( "rf_setup = " );
  Serial.println( rf_setup, BIN );
 }
 void loop() {

 switch1State=digitalRead(switch1Pin);
 switch2State=digitalRead(switch2Pin);
 delay(70);
 lcd.setCursor(0,0);
 lcd.print("Constantes:
");Serial.println(PosMenu);
 lcd.setCursor(0,1);

 //Imprimir en pantalla dependiendo de la
posición que estemos en el menú.
    switch(PosMenu){
     case 0:
     lcd.print("Temperatura: 36C    ");
     break;
     case 1:
     lcd.print("Presion: 120        ");
     break;
     case 2:
     lcd.print("Pulso: 80           ");
     break;
     }

 //Cambio de menú dependiendo del
botón izq. o dcho. que pulsemos.
     if(prevSwitch1State!=switch1State
or prevSwitch2State!=switch2State){
       if    (switch1State==LOW    and
PosMenuMax>=PosMenu){
       PosMenu++;
       prevSwitch1State=switch1State;
       }
       if    (switch2State==LOW    and
PosMenuMin<=PosMenu){
       PosMenu--;
       prevSwitch2State=switch2State;
       }
     }

  //Parte radio - Recibir
  byte c;
  if (Mirf.dataReady()) {
   Mirf.getData(&c);
```

```
char letter = char(c);
if (letter != '~')
{
  message = String(message + letter);
}

else
{
  Serial.println(message);
  message = "";
}
}
delay(80);
//Parte radio - Enviar
  Mirf.send((byte *) &datoenviado);
while (Mirf.isSending())
{
}
```

Anexos.

# Bibliografía

[1] DOUTIEL, FERNANDO (2015). "Guía del Arduinomaníaco: todo lo que necesitas saber sobre Arduino", *Revista electrónica Xataka.* Consultado el 15 de septiembre de 2016 en http://www.xataka.com/especiales/guia-del-arduinomaniaco-todo-lo-que-necesitas-saber-sobre-arduino

[2] PROMETEC (2016). *"Ejemplos, documentación y piezas."*. Consultado el 25 de septiembre de 2016 en http://www.prometec.net/

[3] SIRF (2005).*" NMEA Reference Manual". Manual técnico.* Consultado el 1 de Octubre de 2016 en *https://www.sparkfun.com/datasheets/GPS/NMEA%20Reference%20Manual1.pdf*

**Listado de páginas web consultadas en Internet[48]**

Tutorial del MIT sobre Arduino:

https://www.youtube.com/watch?v=U2fUvCFNyI0

Tutorial Arduino:

http://www.ardumania.es/

http://www.prometec.net/indice-tutoriales/

The Ultimate FM Transmitter (Long Range Spybug)

http://www.instructables.com/id/The-Ultimate-FM-Transmitter/

Infrared Pulse Sensor:
https://www.youtube.com/watch?v=psTa5ZrqAyo&list=PLAZTGLBL-zgT9XyNkPcQd9cG5xNJUW6S1&index=7

GPS Ublox: NEO-6M módulo GPS con MATLAB por USB: http://hetpro-store.com/TUTORIALES/gps-ublox-neo-6m-modulo-con-matlab/

Tutoriales GPS Arduino:

http://playground.arduino.cc/Tutorials/GPS

http://playground.arduino.cc/InterfacingWithHardware/Nrf2401

Tutoriales de C++:

http://codigofacilito.com/courses/c-plus-plus

[48] (Todas estas páginas han sido consultadas durante el periodo que ha durado la elaboración del TFG, última consulta el 30/10/16):

Bibliografía.


http://www.cplusplus.com/doc/tutorial/

http://www.tutorialspoint.com/cplusplus/

Librería Tiny Gps: http://arduiniana.org/libraries/tinygpsplus/

Tienda Online con tutoriales de Arduino: http://thingnovation.com/

Internet of things - beyond our current imagination, Ashkan Fardost: https://www.youtube.com/watch?v=sgMG7zRrcPk

Noticias tecnológicas de defensa: http://www.infodefensa.com/es/

Conectar Arduino con su Teléfono Android por Bluetooth creando su propia App: https://www.youtube.com/watch?v=j-cUDfmtq5g

Blog de proyectos electrónicos: http://hackaday.com/

Módulo HM-10: http://wiki.makespacemadrid.org/index.php?title=M%C3%B3dulo_HM-10